Feasibility of Onboard Processing of Heuristic Path Planning and Navigation Algorithms within SUAS Autopilot Computational Constraints

THESIS

MARCH 2014

Charles J. Neal, Captain, USAF

AFIT-ENV-14-M-44

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-14-M-44

FEASIBILITY OF ONBOARD PROCESSING OF HEURISTIC PATH PLANNING AND NAVIGATION ALGORITHMS WITHIN SUAS AUTOPILOT COMPUTATIONAL CONSTRAINTS

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Systems Engineering

Charles J. Neal, BS

Captain, USAF

March 2014

AFIT-ENV-14-M-44

FEASIBILITY OF ONBOARD PROCESSING OF HEURISTIC PATH PLANNING AND NAVIGATION ALGORITHMS WITHIN SUAS AUTOPILOT COMPUTATIONAL CONSTRAINTS

Charles J. Neal, BS
Captain, USAF

Approved:

//signed//	14 March 2014
Dr. John Colombi, AFIT/ENV (Chairman)	Date
//signed//	14 March 2014
Dr. David Jacques, AFIT/ENV (Member)	Date
//signed//	14 March 2014
Maj Brian Stone, AFIT/ENS (Member)	Date

Abstract

This research addresses the flight path optimality of Small Unmanned Aerial Systems (SUAS) conducting overwatch missions for convoys or other moving ground targets. Optimal path planning algorithms have been proposed, but are computationally excessive for real-time execution. Using the Arduino-based ArduPilot Mega Unmanned Aerial Vehicle (UAV) autopilot system, Hardware-in-the-Loop (HIL) analysis is conducted on default mobile target tracking methods. Designed experimentation is used to determine autopilot settings that improve performance with respect to path optimality. Optimality is characterized using a weighted combination of stand-off range and aircraft roll-rate. Finally, a state-based heuristic navigation strategy is designed, developed, and tested that approximates optimal path solutions and can be used for real-time execution. A 66% improvement in mean performance is achieved over default target tracking methods. Finite state machine improvements are found to be statistically significant and it is concluded that heuristic strategies can be a viable approach to realizing near-optimal SUAS flight paths utilizing onboard processing capabilities.

To Megan, for her unwavering support

Table of Contents

Page

List of Figures

Page

List of Tables

Page

List of Equations

Page

FEASIBILITY OF ONBOARD PROCESSING OF HEURISTIC PATH PLANNING AND NAVIGATION ALGORITHMS WITHIN SUAS AUTOPILOT COMPUTATIONAL CONSTRAINTS

I. Introduction

Background

As unmanned systems technology decreases in both size and cost, the range of applications grows. In particular, use of Small Unmanned Aerial Systems (SUAS) has seen a disproportionately high amount of growth as the affordability of subcomponents has allowed for an increase in availability to probable markets. Applications include, but are not limited to defense, agriculture, law enforcement, and numerous commercial endeavors. Yet no matter how complex or adaptive the payload, the design of any truly purpose-built SUAS must be considered with respect to all subsystems and their contribution to the desired mission. This design focus holds especially true for the navigation logic of the autopilot as increased autonomy is frequently considered an enabler for proposed applications, particularly those in the defense realm.

To that end, multiple research efforts at the Air Force Institute of Technology (AFIT) have culminated in algorithms that provide theoretical aircraft control for various missions extending beyond the existing functionality of most available autopilots. One such effort is the development of an optimal path planning algorithm for tracking and surveillance of a moving ground target [1]. Heuristic variants of these calculations have been suggested with the potential to be implemented onboard existing SUAS autopilots allowing for real-time, autonomous execution. This work has been proposed and supported by the Air Force Research Laboratory (AFRL) as an enabling capability for

convoy overwatch using SUAS. While this mission may be partially achievable with basic manipulation of autopilot waypoints, a more custom approach to navigation logic, capable of implementation onboard the air vehicle, provides potential for increased flight path optimality.

Statement of Problem

The convoy overwatch scenario proposed by AFRL involves the use of a field-deployed SUAS to autonomously track and provide intelligence, surveillance, and reconnaissance (ISR) on mobile ground vehicle maneuvers. Current SUAS convoy ISR operations require a pilot to monitor the air vehicle and a sensor operator (often the pilot in a dual role) to command the payload. Typically, these are continuous functions for the duration of the mission, both of which are required in order to keep the sensor on target and the air vehicle within specified flight parameters. Rather than placing a constant workload on one or more individuals, the proposed functionality would allow for autonomous execution of the mission by the SUAS. A single operator could launch the air vehicle, input flight parameters (target of interest, desired stand-off distance, and sensor angles), and focus attention elsewhere until recovery is required.

While this autonomy may be partially realized using dynamic waypoint capabilities that already exist on some SUAS autopilots, past work suggests that an optimized path planning approach may result in significant performance increases in terms of target tracking and air vehicle endurance [2]. Current AFIT research by Livermore seeks to design such an approach utilizing a cost function to minimize air vehicle control effort and maximize time spent with the sensor at a given stand-off

distance [1]. However, there are indicators that implementation of such a function can be infeasible with the resources available onboard a SUAS autopilot [3]. True optimization functions typically require high computing times while real-time execution of the proposed missions will require multiple iterations per second. Other past research efforts have addressed this issue and suggest that under certain circumstances, optimal routing algorithms can be sufficiently mimicked using more manageable strategies [4]. In order to achieve the desired performance, the specific challenge is the design and implementation of a heuristic approximation of the proposed optimization algorithm that is capable of real-time, autonomous execution onboard the SUAS.

Research Objective

The primary objective of this research effort is the implementation of a heuristic, autonomous autopilot flight mode that replicates, to the best extent achievable, the performance of an actual path planning optimization function designed for the proposed convoy overwatch scenario. Design iterations of this mode are flight tested with the provided results focused on the achieved versus optimal performance and the feasibility of integration into operational systems. The intent is to provide information and analysis sufficient for AFRL to make informed decisions on continuation of future research and development efforts in the field of optimized tracking using SUAS. Additionally, implementation is achieved in a manner that considers the architecture best suited for enabling future integration of customized autonomous navigation functions.

Investigative Questions

Work focuses on answering the following investigative questions sequentially in order to achieve the primary research objective with a build-up approach facilitated by flight test resources available to AFIT:

1. What is the target tracking and flight path performance of the SUAS when using a basic follow-me mode? The follow-me mode describes a very simple approach in which the autopilot is fed a series of target location coordinates at a fixed frequency and updates its current navigation waypoint to match. Most available SUAS autopilots have this capability and it serves as an intuitive starting point for most target tracking missions. The reason for characterizing tracking and navigation in this mode is that it serves as a comparative baseline for evaluating performance of any other tracking algorithm. Note that for this effort, qualitative reference to performance or optimality of any flight path is based on the similarity of the path to that which could have conceivably been achieved under identical conditions as calculated by Livermore's optimization algorithm. Details on the measures of optimality are discussed in Chapter 2.

2. What is the best path performance achievable by the adjustment of existing or readily accessible navigation control without implementation of state responsive logic? The process by which this question is answered is intended to make existing navigation functionality achieve the most optimal flight paths possible with regards to ground target tracking. It is important to ascertain these settings before proceeding to evaluation of states within which varied control logic may be appropriate.

3. What is the achievable SUAS flight path optimality using a state-based, heuristic approximation of the optimization strategy? The intent of this question is characterization of the attempted heuristic path planning strategy with respect to baseline, adjusted, and true optimal performance.

4. What is the feasibility of implementing heuristic ground target tracking logic that is capable of real-time execution onboard a SUAS autopilot? This question is designed to answer the overarching research objective based on answers from all preceding questions. The feasibility analysis is formulated based on an assessment of the achieved performance during SUAS flight test events designed to replicate the convoy overwatch scenario.

Assumptions and Constraints

The proposed convoy overwatch scenario has a wide potential range of application and complexity, varying from straight line path following to highly diverse road networks with high levels of variance in vehicle speed, direction, and altitude. For this research, a set of assumptions is made to facilitate the planning of achievable experiments with meaningful results than can be conducted within the constraints of equipment and range time available to AFIT. The baseline scenario is that of a SUAS providing overwatch for a ground control station (GCS) located on a mobile ground vehicle of known global positioning system (GPS) coordinates. The actual path driven by the ground vehicle for all tests associated with this effort is shown in Figure 1. This route was selected based on range availability and safety approvals.

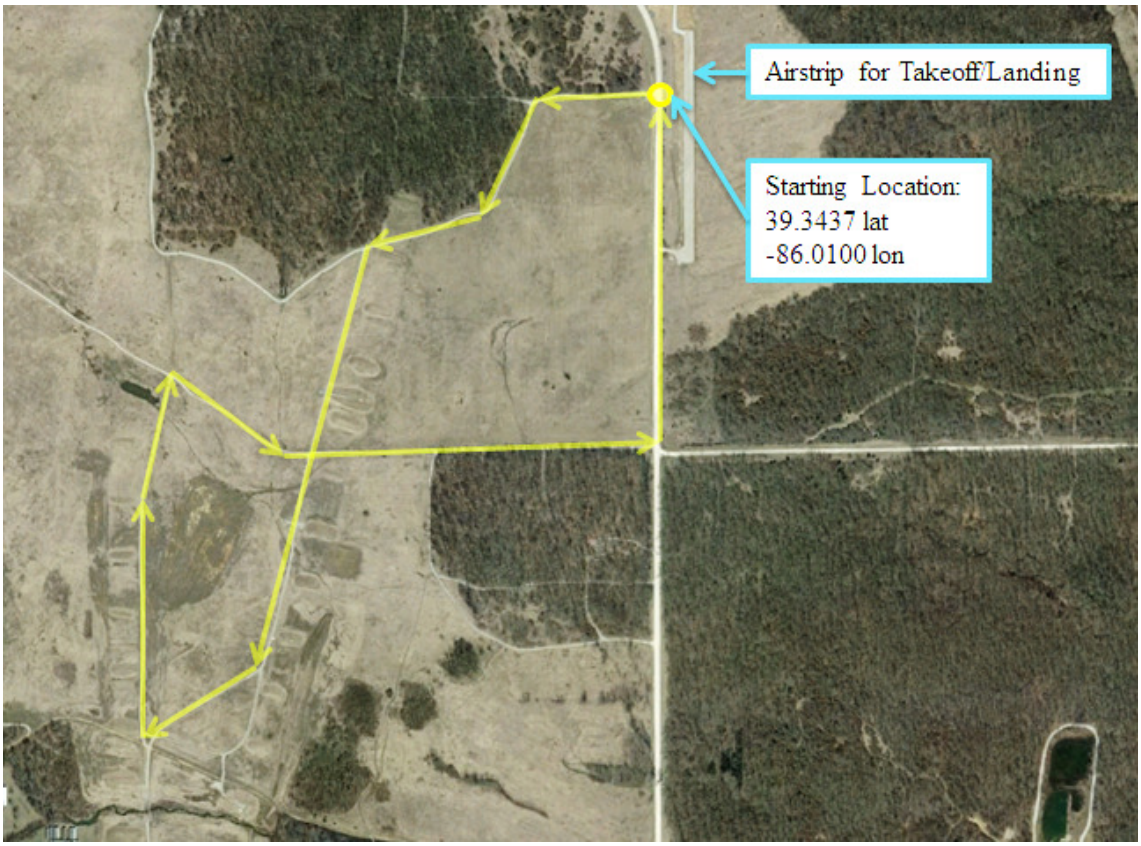

Figure 1: Ground Vehicle Path Used For Testing

For any ground tracking scenario, it is assumed that a well-designed system is one in which the ground speed of the air vehicle while commanding its optimal cruise throttle setting into maximum expected wind conditions is also the maximum ground speed that may be reached, either momentary or steady-state, of the ground target in question. This speed is characterized for the SUAS used during experimentation and the maximum speed of the ground vehicle is constrained accordingly. Failing to make this design choice allows for states in which the ground vehicle may simply outrun the air vehicle. Additionally, this research assumes that altitude variance in the ground path is negligible

and, based on safety concerns, all SUAS flights are performed at a fixed altitude of 150 meters above ground.

Regarding the air vehicle specifically, one constraint placed on the research effort is the use of waypoint navigation instead of fly-by-wire navigation. It is assumed that any autopilot potentially fielded for target tracking missions is capable of waypoint navigation, including the capability to update waypoints dynamically and perform a fixed loiter should it arrive at a waypoint without receiving any updates. All developed logic uses point navigation as opposed to a fly-by-wire approach which would involve direct control of flight conditions such as bank, pitch, and heading.

Finally, it is assumed that any SUAS to be integrated with the proposed tracking functionality is capable of operating a sensor gimbal to given pointing angles. The algorithm developed generates dynamic target coordinates, but actuation of the gimbal to the desired angle is considered an existing capability of the autopilot or associated peripherals. Furthermore, it is assumed that error in the pointing functionality of the gimbal is negligible and no work is done to provide compensation for pointing inaccuracies.

Overview of Methodology

The first step in this research is the integration of air and ground vehicle telemetry as inputs to MATLAB optimization scripts that will serve as the primary method of generating optimal flight paths using Livermore's proposed cost function. For a given run, the output is an optimal flight path that could have been executed given the physical bounds of the aircraft and environment.

The following step will be development of a heuristic approach to approximating optimized paths that is capable of being integrated in Arduino code and run on the APM without introduction of excessive computing delay. Flight test is designed to evaluate the stock performance, adjusted non state-based performance, and finally modified state-based performance of the SUAS performing a ground vehicle tracking mission. Flight tests are conducted iteratively, with navigation logic for each building on the results of the previous. The goal is to compare achieved optimality, in terms of cost function value, for the above listed flight conditions against each associated optimal solution. Data required for these comparisons includes basic aircraft telemetry (GPS information, aircraft physical state, control effort, and gimbal angles) from real-world flights as well as comparable data from MATLAB generated paths. Differences in performance are used to report on the feasibility of achieving near-optimal target tracking missions with high levels of autonomy using existing autopilot computing resources. Additionally, discussion is provided on the architecture required to implement customized flight modes onboard the APM.

Thesis Overview

This chapter provides a brief background on SUAS, description of the motivation for integrating heuristic tracking strategies onboard SUAS autopilots, discussion of the specific research tasks to be addressed, and an overview of the equipment and methodology used. Chapter 2 examines literature and past work relevant to this effort providing validation of the equipment selection, problem statement, and experimentation methodology. In addition, further discussion is given to the expectation of performance

differences for SUAS missions under optimal, near-optimal, or non-optimal planning methods. Chapter 3 provides a more in-depth look at the test methodology with greater emphasis on specific test events. Chapter 4 presents the software design and the results of the research efforts built on data that have been collected and processed. Chapter 5 concludes the thesis and discusses implications of this work as well as recommendations for future efforts.

II. Literature Review

Chapter Overview

The literature review is intended to provide a synopsis of research efforts and findings that are relevant to, or have culminated in, the challenge of characterizing SUAS heuristic tracking algorithm performance. While the motivating requirement for the current research effort has been proposed by AFRL, it is appropriate to mention that other sources allude to the current or future need for optimized ground tracking capabilities. The United States Air Force Unmanned Aircraft Systems Flight Plan 2009-2047 lists many UAS currently used in deployed environments as well as generic capabilities of UAS in different size classes [5]. Only two aircraft specifically include convoy overwatch in their lists of capabilities, the MQ-1 Predator and the MQ-9 Reaper. However, in its coverage of future applications of SUAS, the UAS Flight Plan lists close-in ISR, personal ISR, and auto-sentry. These missions will likely include (as a subset) autonomous tracking of a ground target, whether friendly or hostile. In a 2011 RAND Corporation report to the US Army, Peters et al. discuss the technical and operational feasibility of overwatch missions by UAS [6]. They argue that large UAS present the most technically feasible options for convoy overwatch but claim that operational feasibility is highly constrained by the tasking complexity and low availability of this aircraft class. Their final assertion is that feasibility would be positively impacted if miniaturization of technology enabled vehicle overwatch to be performed by smaller, cheaper UAS.

For the remainder of this chapter, topics specific to the current research are addressed. Coverage is given to the expected benefits of optimized routing followed by

the specifics of current path planning research efforts. Past work is discussed on the subject of approximating optimization algorithms in real-time. Finally, research is examined that discusses performance characterization of small UAS with regards to metrics and utilities relevant to validating the experimentation methodology of this research effort.

Flight Path Optimization

Characterizing the performance implications of approximated optimal path planning solutions warrants discussion of three key areas. First is the expected impact of optimization on SUAS performance. Second is the work currently proposed for achieving the overwatch mission in question. Last is the challenge of approximating optimal solutions in a heuristic manner. Prior work on each of these topics is examined.

Effects on SUAS Performance

While the current research effort characterizes performance primarily with respect to path planning, it is important to note that previous work provides preliminary indicators of other potential benefits. Research conducted by Lazano examines performance of SUAS autopilot control loops parameterized to optimize flight endurance and optical sensor effectiveness [2]. A predicted 33% increase in flight endurance is achieved by altering pitch-from-altitude control loop settings. The performance difference is attributed to the amount of work required of these control loops when deviating from steady level flight conditions, either intentionally or unintentionally, suggesting that the best way to optimize endurance is to minimize altitude holding efforts by the aircraft. It follows that the cost function to be utilized in the current research,

which seeks to minimize roll rate and consequently altitude holding effort, can reasonably be expected to have a positive impact on mission endurance.

Lazano continues by examining the surveillance efficiency of his missions. It is suggested that considering navigation waypoints separate of sensor aimpoint results in decreased surveillance effectiveness and optimality of the flight path. He asserts that implementing a gimbaled sensor with path planning based on footprint location may be the most significant contributors to ISR effectiveness for SUAS. In his research conclusion, with specific regards to "loiter surveillance and moving-target surveillance," Lazano recommends that "additional research should be conducted to determine improved persistence settings for respective surveillance methods" [2, pp. 95-96].

Current Efforts

AFIT research has been conducted to directly address the convoy overmatch problem proposed by AFRL. This effort is presented by Livermore where he proposes a dynamic path optimization strategy designed to minimize both error in SUAS distance from the ground target and SUAS control effort [1]. This strategy begins by defining a function which characterizes the cost, J, of any given SUAS flight. This function is defined in Equation 1 [1].

> The cost function aims to minimize the weighted sum of the control and slant range (SR) error. The cost function represents the desire to keep the UAV a certain distance from the ground vehicle while using the minimum required control. In [Equation 1], the first term penalizes deviation from desired slant range and the second term penalizes the control. Both the slant range and control terms are normalized relative to constant values so that the two terms can be equally weighted relative to each other. [1, p. 36]

Equation 1

$$J = \int_{t_0}^{t_f} \left[\alpha \left(\frac{SR(t) - SR_{desired}}{SR_{desired}} \right)^2 + (1 - \alpha) \left(\frac{u(t)}{u_{max}} \right)^2 \right] dt$$

Where:
$t_f = final\ flight\ time$
$t_0 = initial\ flight\ time$
$\alpha = relative\ weight$
$SR = slant\ range$
$u = roll\ rate$

After establishing this cost function, Livermore develops a MATLAB function that accounts for the path driven by a ground target, weather conditions, the starting location of the SUAS, the desired slant range, the u_{max} specific to the SUAS, as well as speed and turning characteristics specific to the SUAS. With these inputs, the function attempts to identify the most optimal flight path that could have been executed. The selected path is defined at that with the lowest associated cost [1]. An example of Livermore's path generation based on real world ground vehicle and weather information is show in Figure 2.

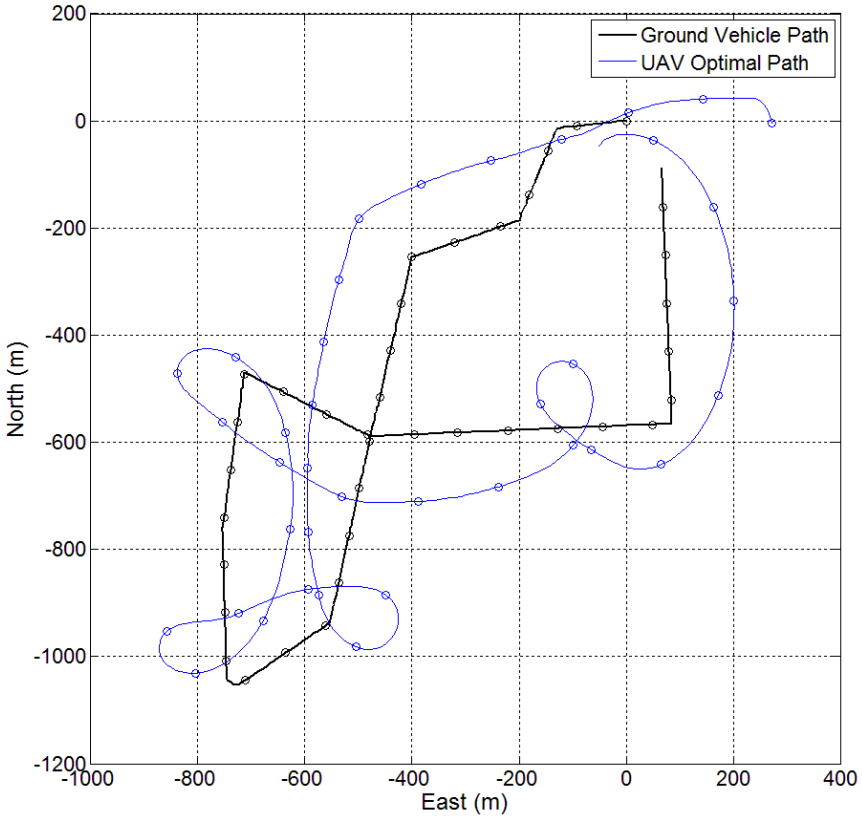

Figure 2: Example Optimal Path Generation

Approximations of Optimal Solutions

In early AFIT optimization work, Zollars proposes a dynamic optimization algorithm that determines the best route for a SUAS attempting to place a sensor footprint on a target of known location and velocity [7]. While the motivation for his work is different than that of the current effort, he arrives at a computationally intensive optimization algorithm similar to that being evaluated at present. Implementation of Zollars' work is attempted by Terning, who works to "specifically look at heuristic, iterative techniques which can quickly calculate flight path solutions, implement these

solutions on actual UAV systems, and validate the algorithm through flight tests" [3, p. 3]. Terning concludes that the amount and variance in execution time makes Zollars' technique infeasible for direct application in real-time circumstances:

> Because the code execution time proved unpredictable, it proved impossible to extrapolate out the future position of the aircraft to a point where the flight path commands would actually be executed. If, for example, we knew with relative certainty that it would take 10 seconds to compute an optimal flight path, we could effectively extrapolate the future location of the UAV, and optimize for that point. If, however, the calculation time is unpredictable and highly variant, no prediction can be made. The other option would be to force a return after a certain number of seconds. This would essentially guarantee an erroneous result of unknown tolerance if the optimization routine was exited prematurely, so this option was abandoned. [3, p. 20]

Terning's final solution is an iterative approach that evaluates various coordinates along the vector of the ground target based on present information about both the target and the air vehicle. When the calculated time-of-arrival becomes equal for both entities (or nearly equal as predefined by a threshold parameter), the evaluated location becomes the new navigation point for the SUAS. The GCS software executes this calculation repetitively, each time updating the navigation point. Terning demonstrates his heuristic approach using a hardware-in-the-loop (HIL) simulation and provides strong evidence that an iterative approximation of an optimization based on cost functions can be achieved in real-time with worthwhile results.

A similar strategy is seen in research presented by Boire, who builds on the work of Seibert et al. and attempts to achieve an implementation of the aforementioned rover-relay architecture [4]. Boire notes that for an instantaneous set of aircraft states (both rover and relay SUAS) it is a simple midpoint calculation to determine the optimal location at which the relay should be positioned. However, when attempting to account

for future states based on the motion of both SUAS, the optimization function becomes complex enough that an approximation is the most feasible approach to real-time implementation. He arrives at a strategy of repetitively calculating and commanding an instantaneously optimal solution, including a future position compensation factor for the rover SUAS. The cyclical nature of the approach makes it similar to Terning's work. However Boire's method differs in that the calculation itself is not recursive. The strategy is implemented in the proposed GCS software and demonstrated in simulation. Findings indicate that his solution is able to achieve a range increase for the rover SUAS close to that expected of the optimal solution, providing further evidence that heuristic approximations can effectively emulate their optimal counterparts if designed properly.

Sensor Time-on-Target

In addition to the development of SUAS path planning strategies, it is of equal importance for the current research effort to validate achieved performance. The primary challenge is ensuring that the sensor maintains persistent coverage of the ground target in question. For this research, it is sufficient to quantify the percentage of flight time during which the sensor field of view encompassed the target.

Welborn encounters the same issue in his research attempts to quantify achievable ISR for the Raven SUAS [8]. His approach builds on a basic MATLAB script originally built by Lozano for visualizing a sensor aimpoint and footprint [2]. Welborn modifies the script to characterize dynamic flight telemetry and provide statistical output for time on target. Because his work utilizes real telemetry files and hard-coded sensor angles, the generated time on target is theoretical for a real world flight, which helps account for

sensor mounting error that may be present in the actual video. Additionally, adjustment of inputs allow for performance analysis of alternative sensor configurations without requiring extra flights. Welborn's utility is used for calculating achieved time on target for all flight tests executed in the current research effort. The generated visualization of sensor aimpoint and footprint assists in characterizing flight conditions contributing to gimbal performance. Modifications to the utility include telemetry input format, dynamic sensor angles from telemetry (to account for a gimbaled camera), and dynamic ground target location (to account for a moving target). Figure 3 shows an example ISR flight visualization generated using Welborn's utility for a fixed body camera.

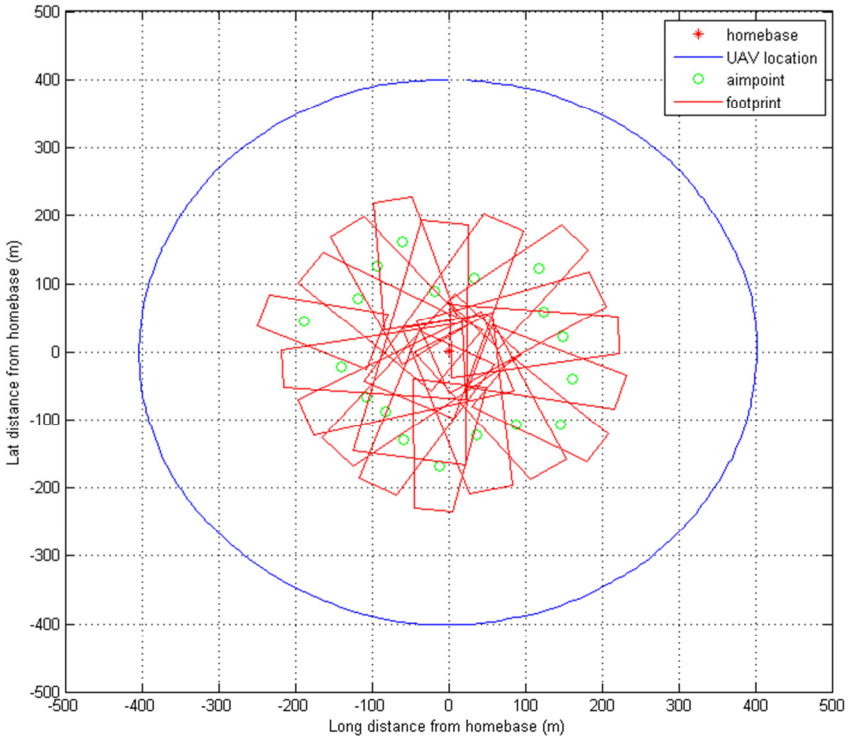

Figure 3: Welborn Example Flight Path with Sensor Aimpoint and Footprint

Summary

The literature review examines prior work that has culminated in, contributed to, or provided justification for the current research effort. Initial focus is given to documentation supporting the requirement for an optimized mobile ground target tracking function. SUAS work at AFIT is then reviewed to justify some of the key equipment selections made prior to executing flight test. Research on the potential effects of optimized path planning is discussed that further supports the thesis motivation. This is followed by a more thorough examination of efforts to optimize the convoy overwatch mission as well as past work to approximate similar path planning functions. Finally, coverage is given to supporting work providing performance validation and analysis utilities directly relevant to the experimentation portion of this research.

III. Methodology

Chapter Overview

The methodology chapter describes the process used to answer the stated investigative questions associated with the research objective. Those questions are as follows:

- What is the target tracking and flight path performance of the SUAS when using a basic follow-me mode?
- What is the best path performance achievable by the adjustment of existing or readily accessible navigation control without implementation of state responsive logic?
- What is the achievable SUAS flight path optimality using a state-based, heuristic approximation of the optimization strategy?
- What is the feasibility of implementing heuristic ground target tracking logic that is capable of real-time execution onboard a SUAS autopilot?

Each of the investigative questions is designed to augment its predecessor, cumulatively arriving at a feasibility assessment regarding SUAS autonomous mobile target tracking. The determination of feasibility is justified by characterizing the spectrum of achievable performance and recording how heuristic approximation compares to worst and best case scenarios.

Documentation of the methodology begins with a discussion of the materials and equipment to be used for the research effort. This is followed by examination of the procedures followed in order for experimentation to provide the data required to analyze current performance and design an improved navigation strategy.

Materials and Equipment

The traditional components of a SUAS include the air vehicle, payload, ground control station, communications, launch and recovery hardware, and ground support equipment [9]. These components can be divided into various subcomponents unique to the system and its mission. The conclusions of this research effort are based primarily on data gathered from flight test. For that reason, it is appropriate to review the components and subcomponents of the SUAS used in testing that most directly impact or constrain the data collected. Those components include the air vehicle, autopilot, ground control station, and sensor gimbal.

Air Vehicle

The air vehicle used for this testing is the Sig Rascal 110. This aircraft is a commercial-off-the-shelf (COTS) hobbyist RC aircraft that has been modified for use as an AFIT SUAS test platform. Modifications include upgrades to battery and power-plant for increased reliability and endurance, as well as installation of an autopilot. The Rascal is conducive to AFIT flight research due to its availability and current status as an approved airframe for USAF test on the Atterbury range. Figure 4 shows the Rascal in use during flight test at Camp Atterbury. See Appendix A for detailed specifications.

Figure 4: Rascal SUAS

Ground Vehicle

The ground vehicle used for all flight test associated with this effort is a military HMMWV troop carrying vehicle. This selection is based on safety approval considerations and range availability. As configured, the vehicle allows for a driver and ground station operator in the cab of the vehicle with the safety pilot seated in the rear to maintain view of the SUAS. The HMMWV used for testing is shown in Figure 5.

Figure 5: Ground Vehicle

Autopilot

Many COTS SUAS autopilots are available on the market with wide variance in cost and capability. The autopilot currently in use for AFIT research is the ArduPilot Mega (APM) version 2.5. The APM is built as a variant of the Arduino electronics prototyping board. In addition to being low-cost, the APM has been selected because it is an open source platform. All firmware being run onboard is available in community repositories rather than being treated as proprietary to an originating designer, which makes the APM conducive to research efforts requiring custom code.

The APM is similar in size, computing power, and flight functionality to those autopilots currently used in many fielded systems [9]. This similarity helps ensure

transferability of the results, as the proposed convoy overwatch scenario is primarily a defense application. The APM is designed to operate a variety of ground or air vehicles based on the firmware being run. For this research effort, the ArduPlane Arduino sketch is used, which is designed primarily for powered, fixed wing aircraft. Peripherals to the APM include a transceiver for telemetry and real-time control, a GPS receiver, a barometric pitot-static unit for airspeed and altitude measurements, and a magnetometer for heading measurement augmentation. Figure 6 depicts the APM with key components labeled [10]. Reference Appendix B for detailed specifications.

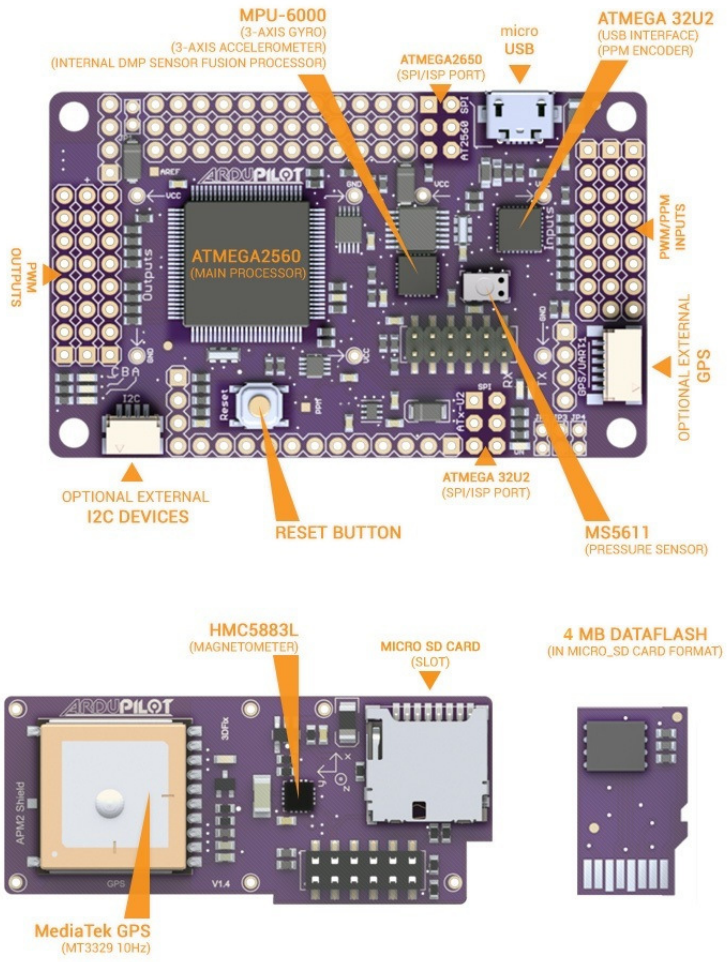

Figure 6: ArduPilot Mega

Ground Control Station

The GCS selected includes a laptop running Microsoft Windows, a telemetry transceiver matching that onboard the aircraft, and the APM Mission Planner software. This software is also open source and provides the functions required to monitor the SUAS in real-time and provide any required control updates. Like the APM, Mission Planner is highly representative of ground control software found in many fielded systems. The similarity contributes to the utility of findings while the fact that it is open source allows for modification of functionality. In addition to the standard GCS configuration, a GPS receiver is integrated with the laptop to provide information on the ground vehicle location and velocity while moving. A screenshot of the Mission Planner software used for this research effort is shown in Figure 7. For details on the specific GCS setup used for this effort, reference Appendix C.

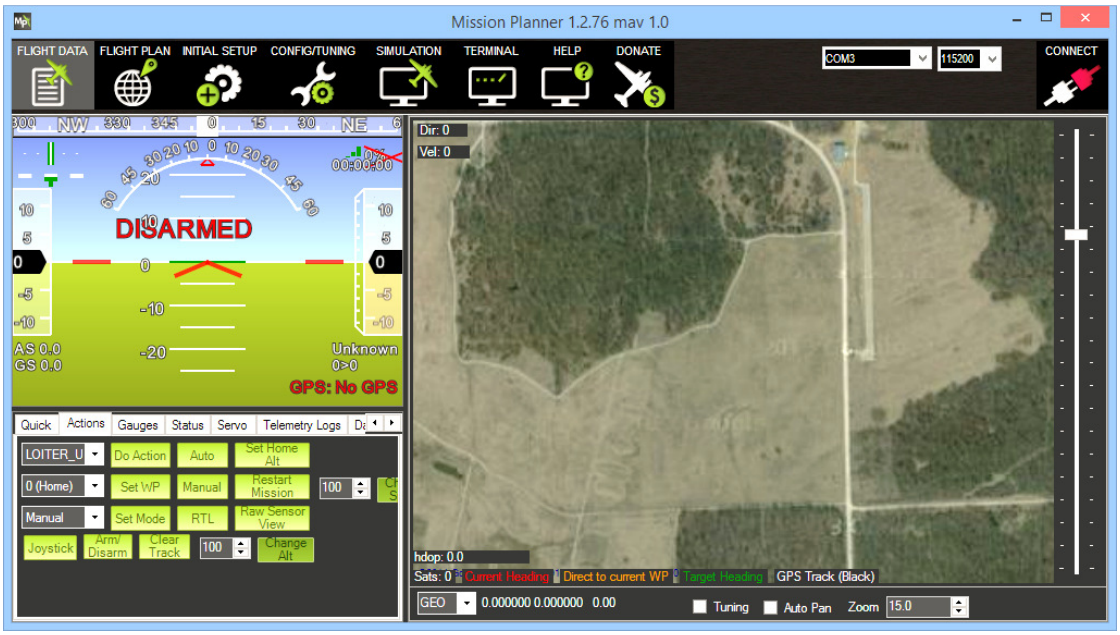

Figure 7: Mission Planner Screenshot

Sensor Gimbal

The payload integrated onboard the Rascal is mounted on a two-axis (pan-tilt) gimbal comprised of all COTS components with stabilization actuation provided by the autopilot. The frame is built on two RC servos. The pan servo allows for ±180° rotation from its center position. The tilt servo is capable of +10° and -90° rotation from the horizontal plane of the SUAS. For this effort, all servo commands are generated directly by the APM. Minor code modifications allow the autopilot to actively update look angle (and subsequent servo positions) while flying in a dynamic ground vehicle tracking mode. Chapter 4 provides a more detailed discussion of all firmware modifications.

The camera used is the HackHD board camera. The HackHD is a high-definition (1080p) color camera with a standard lens mount so that the optics can be altered to meet specific mission needs. In addition, the camera supports onboard recording of video to a micro-SD flash memory card which allows for post-processing of full quality video and makes real-time transmission optional for testing purposes. Figure 8 shows the integrated camera and gimbal system mounted to the Rascal in flight configuration. Reference Appendix D for detailed payload specifications.

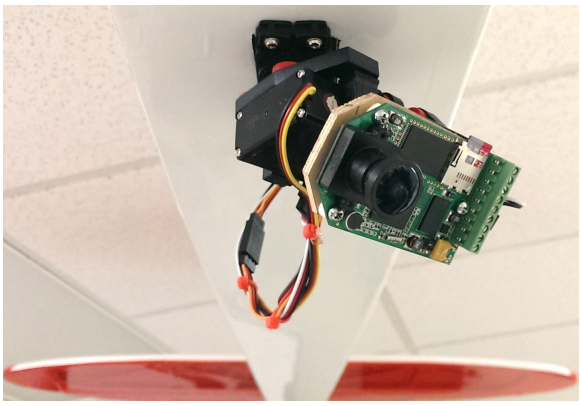

Figure 8: Gimbal with Video Camera

Range Support and Flight Preparation

All SUAS flight tests for this research are conducted at the SUAS airstrip located at Camp Atterbury Joint Maneuver Training Center in Indiana. All flight tests require AFIT support in scheduling range time and providing necessary coordination with Camp Atterbury. In addition, AFIT policy dictates that a Form 5028 be submitted prior to any flight testing. This form outlines specific equipment configurations and actual flight test points to be executed. Approval of the Form 5028 may only be attained after presentation in both a Test Review Board (TRB) and a Safety Review Board (SRB).

Hardware in the Loop Simulation

In addition to flight test data collected on real-world equipment, the effort leverages the APM capability to execute some of the flight test in a hardware-in-the-loop (HIL) simulated scenario. This allows for collection of a higher number of test points than otherwise possible with fewer safety and logistical considerations. HIL simulation works by connecting the APM to the GCS computer over a serial port. In addition to the Mission Planner software, a simulated flight environment, FlightGear, is run using a model version of the Rascal airframe. FlightGear uses a model called JSBSim for 6-degree-of-freedom flight dynamics simulation as well as aircraft parameter definitions [11]. Figure 9 depicts the communications architecture for running HIL simulations. Note that in this configuration, the APM is running all navigation logic in an identical fashion to real-world flight. Only processes responsible for reading sensor data are aware that state information should be obtained from the serial port rather than actual sensors. Because of the object oriented nature of the firmware, the source of this information is hidden when passed to navigation processes.

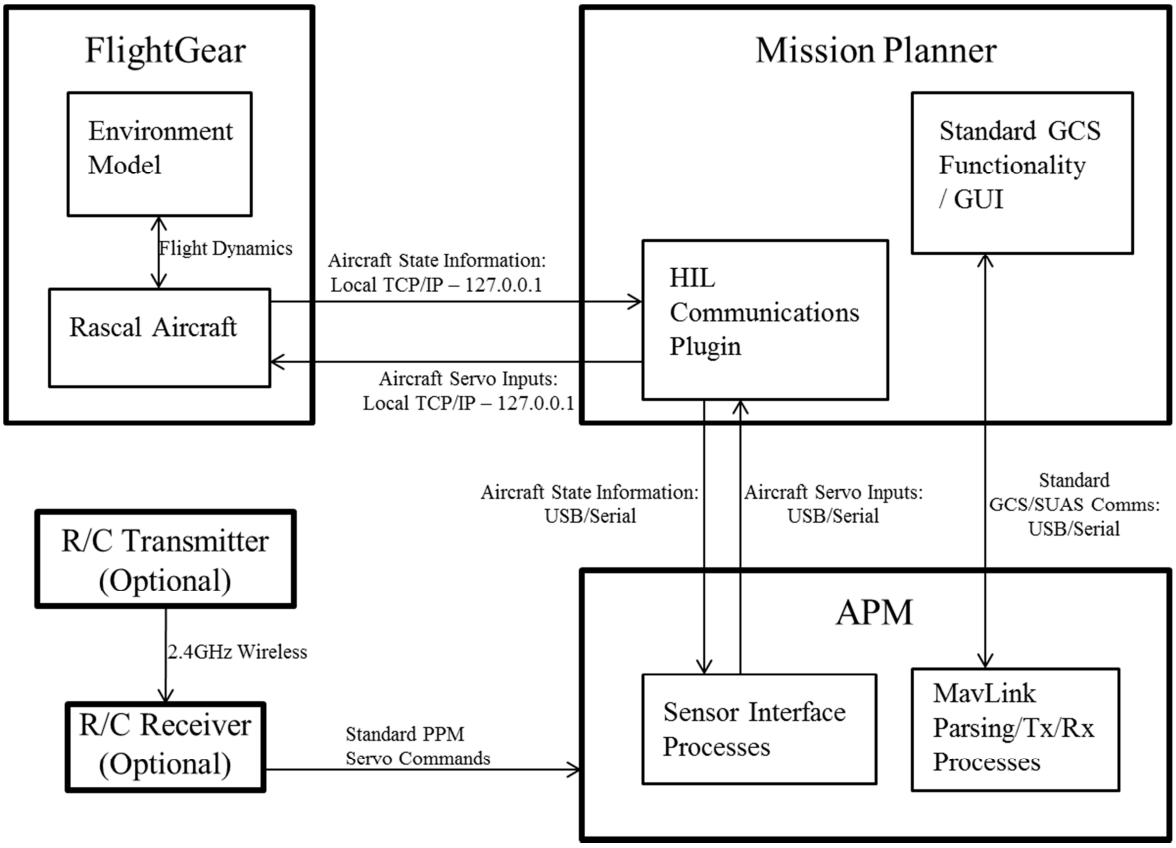

Figure 9: Hardware-in-the-Loop Communications Architecture

In addition to the default HIL configuration described, pre-scripted GPS information can be output on a local virtual serial port, enabling the use of follow-me mode in Mission Planner. Scripting the GPS data to match the profile of the HMMWV executing the selected real-world ground path, as well as using a modeled version of the actual air vehicle being used, allows HIL flights for the effort to match real-world flight performance to a high extent. Reference Appendix E for the definition file used to instantiate the simulated Rascal used in this effort.

Procedures and Processes

The findings of this research effort are formed on data aggregated from real world SUAS flight telemetry as well as emulated flight paths based on real air vehicle characteristics. Work required to collect this data begins with characterization of ground target tracking performance using the unmodified follow-me mode. Performance in this this configuration serves as a baseline. Next, experimentation is done to determine the best settings for all navigation logic pertinent to the proposed path planning strategy. Finally, a finite state machine approach to path planning is constructed with design based on analysis of flights flown at the aforementioned best settings.

Field Data Collection

For flight test (both real-world and HIL simulation), field data is collected in the form of aircraft and ground vehicle telemetry. APM Mission Planner can record certain information directly to telemetry log (TLOG) files for later analysis or simulated re-creation of the flight. For this effort, the TLOG format is used to collect all aircraft data on the GCS laptop. Specific TLOG information of interest includes air vehicle GPS location data, aircraft attitude, gimbal servo outputs, inertial sensor readings, wind conditions, and ground target location data.

For optimal paths calculated in MATLAB, the same data is generated with the exception of ground vehicle location, which must be treated as an input to the function in order for the paths to remain applicable to specific real-world conditions. Air vehicle GPS location, attitude, and gimbal control will all be output by the utility and inertial readings can be derived from aircraft state information.

Follow-Me Flight Test

The initial attempt at a real world, moving ground target tracking effort is achieved with the APM follow-me mode. Follow-me is used as the performance baseline for comparison of all subsequent tracking attempts. This functionality is already partially implemented in the APM and Mission Planner software. The existing function sends a new waypoint to the SUAS at a fixed frequency. The waypoint is simply the location of the GCS (based on a GPS reciever) at the time of the message and does not project to a future intercept point based on velocity. If the aircraft arrives before the waypoint changes, it will enter a loiter about that point.

Flights are conducted with the aircraft placed in follow-me mode and the GCS located on the ground vehicle. The ground profile driven is the pre-selected path introduced in Chapter 1. Air vehicle altitude is fixed at 150 meters as determined by airspace constraints and local terrain. The mission is executed at different SUAS loiter radius settings but the data of interest is that collected at the radius determined to be nearest optimal in subsequent experimentation. Recorded field data includes ground vehicle profile, aircraft telemetry, and ground target video.

Increasing Path Optimality by Experimentation

In order to develop navigation logic in the form of a finite state machine that is responsive to real-time SUAS conditions, the existing performance is examined to identify which states warrant alternative behavior. However, rather than performing this analysis on data from the unmodified follow-me mode, it is first important to adjust any relevant system settings to achieve the most optimal flight paths possible. Flight data

garnered from these settings result in a more appropriate determination of state definitions.

Examination of the existing APM fixed-wing aircraft firmware yields three parameters which directly have roles in the navigation logic that impacts the minimization of roll rate and the maximization of effort to stay at a specified standoff distance. These parameters are the loiter radius itself, loiter range, and navigation point.

Loiter radius is the actual horizontal distance from the ground target point that the aircraft will attempt to maintain. For a fixed target point, this represents a circular loiter. When inside or on the loiter radius, updates to desired heading (which are subsequently fed into lower level control loops) account for the ratio of the current distance from target to the desired distance. The level of effort applied to achieve that distance, which is expressed as the magnitude of change to the desired heading for any one instance of the control loop, directly represents a balance between control effort and slant range.

Loiter range is an additional distance beyond the loiter radius, inside which the SUAS will begin a gradual transition from straight flight towards the target point to circular, tangential flight around the target point. This is a fixed distance, rather than a proportion of the loiter radius, and is designed to allow for smooth entry into loiters with minimal overshoot. Similar to the effects of loiter radius, control effort is directly based on a ratio representing relative position inside the range, meaning that the range itself can impact the optimality of any given flight. Modifications, discussed in Chapter 4, are required to parameterize loiter range, as it is hard coded at 60 meters in the default firmware. Figure 10 demonstrates the role of both loiter radius and range in the APM navigation logic.

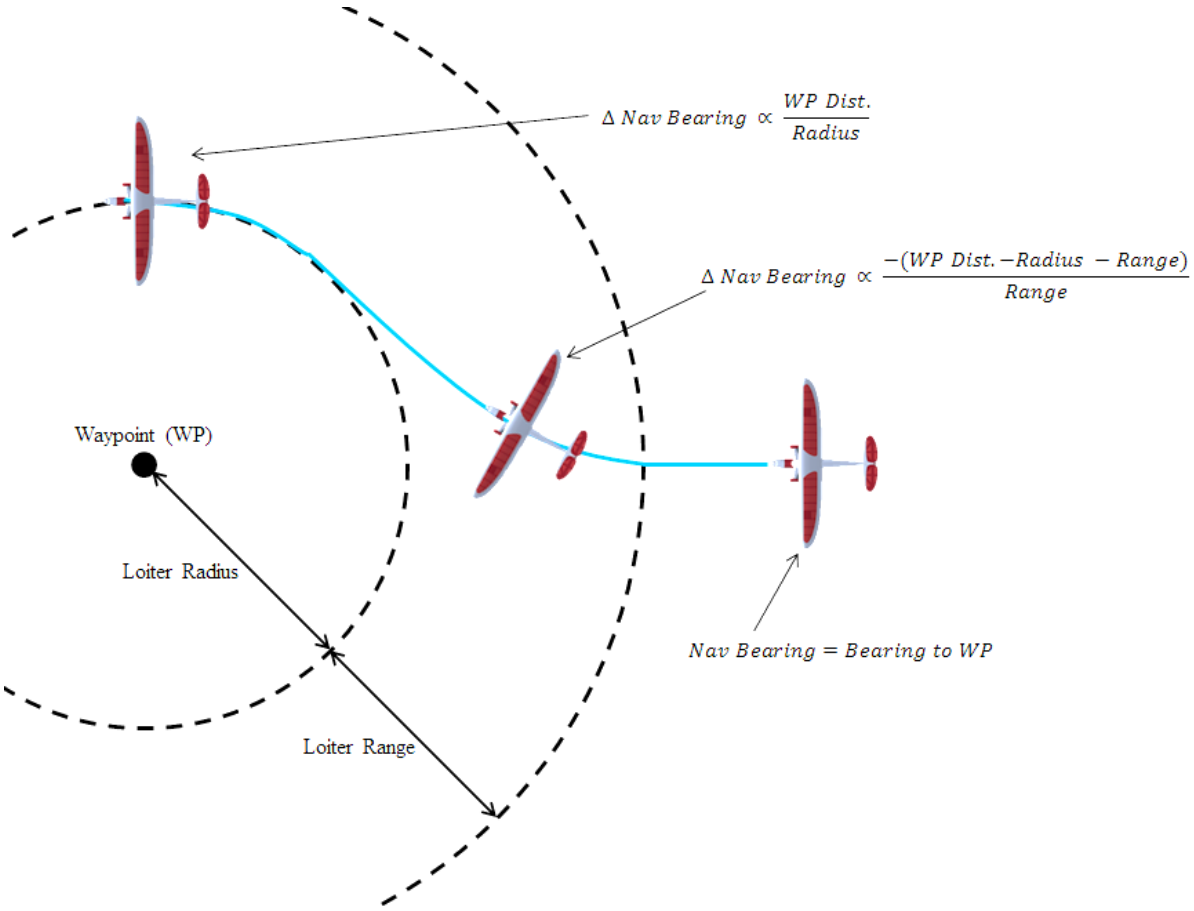

Figure 10: APM Loiter Navigation

Finally, the point to which the aircraft is navigating must be considered. Under normal circumstances, this point is only affected by motion of the ground vehicle. However, Terning's work [3] shows that forward projecting the location of a moving ground target affects the behavior of a SUAS when attempting to intercept a point. Additional APM firmware modifications include the addition of a lead time parameter to account for the possibility of impacting the performance of a ground target tracking mission. For any lead time greater than zero, the SUAS will navigate to a point directly forward of the ground vehicle based on the number of seconds input and the vehicle's velocity.

HIL simulation and range safety limitations are used to identify a safe range of potential settings for all three parameters in question. This allows for design of experiments (DOE) planned tests executed in two stages. The first stage accounts for all three factors tested across their entire range of values to identify which factors have a significant impact on flight cost (optimality) as well as providing an initial assessment of the settings that should be used. A computer generated central composite design (CCD) is used because quadratic effects and two-factor interactions are predicted. The second stage provides finer granularity in a smaller test space to validate the initial findings and arrive at the final recommendations for settings. Again, a computer generated CCD is used.

State-Based Navigation Logic

After completion of a designed experiment to optimize the performance of follow-me mode, a more appropriate analysis of flight data is conducted. This allows for the identification of states in which there is room for improvement in terms of ground target tracking flight path optimality. Analysis of flight data collected at the recommended settings identifies the most noteworthy states with suboptimal performance. To account for these scenarios, a finite state machine is designed that allows the SUAS to execute alternate navigation when the appropriate conditions are met. Modifications made to the APM firmware allow for the implementation of the proposed state machine. After integration, flight test is conducted to verify improved SUAS path performance.

Summary

The methodology chapter examines the work performed to answer the technical investigative questions requisite to produce a feasibility report on achieving efficient ground target tracking missions through heuristic path planning strategies. Initial discussion is on the specific hardware used for testing and how each piece contributes to the research effort. Next, procedures and processes are explained. Focus is given to the necessary order of testing as well as justification for each test, concluding with an overview of the achieved state-based strategy.

IV. Analysis and Results

Chapter Overview

The analysis and results chapter discusses all data that was collected following the methodology outlined in Chapter 3. This discussion begins with an overview of code modifications initially required for the autopilot to perform the convoy overwatch mission. Following is an examination of actual flight data from each of the three phases of test (follow-me, optimal settings experimentation, and finite state machine implantation) to include justifications of associated navigation logic choices. Final examination is focused on additional firmware modifications required to achieve the documented performance.

Initial Firmware Modifications

The initial research phase was used to base-line existing performance, however certain firmware modifications were necessary to enable the experimentation and improvement phases. Changes were made to address two notable shortcomings of the stock ArduPlane firmware. First is a lack of autonomous sensor gimbal control for a moving target. Second is a fixed loiter direction for all modes using loiter-based navigation logic.

Sensor Gimbal Target Tracking

While the APM autopilot has a follow-me navigation function, it proved insufficient to meet the basic requirements of the convoy overwatch mission in its default form. Most notable was the immaturity of the AP_Mount library. The library is used to define the AP_Mount class which, when instantiated as an object by the main ArduPlane

thread (known as an Arduino sketch), represents a sensor gimbal on the aircraft. Methods associated with this class are used for the execution of all sensor gimbal motion.

In its unmodified state, the AP_Mount class is designed to accommodate two basic functions. First is single or multiple axis stabilization about an earth-fixed pointing angle, designed primarily to minimize image motion from the aircraft. The second is a pointing function designed to keep the sensor fixed on a single ground location while the aircraft is in motion. This function is based solely on point-and-click user inputs from the Mission Planner interface, with specialized telemetry link packets for updating commands. Execution of a convoy overwatch mission in this configuration would require a dedicated operator and be inherently inaccurate due to the point-and-click update method.

To accommodate the desired autonomy, modifications were made to allow all ArduPlane processes access to global knowledge of the ground vehicle location. Once the ground vehicle location was available, it could be used to calculate the desired pointing angle within the update_mount_position method in the AP_Mount class. Modifications to this method introduced two input parameters. First was the location of the ground vehicle, replacing the previously internal location calculation. The second was a boolean, used as a flag to inform AP_Mount of the status of follow-me mode where true indicated use of the modified function. This allowed retention of the default functionality should an operator wish to override the gimbal or if follow-me mode was stopped. Figure 11 depicts class diagrams for both the default and modified AP_Mount class with the modified function highlighted. Reference Appendix F for the revised update_mount_position function.

AP_Mount «Default»	
- _ahrs:	AP_AHRS
- _gps:	GPS
- _current_loc:	Location
- _target_GPS_location:	Location
- _mount_type:	MountType
- _roll_idx:	int
- _tilt_idx:	int
- _pan_idx:	int
- _open_idx:	int
- _roll_control_angle:	float
- _tilt_control_angle:	float
- _pan_control_angle:	float
- _roll_angle:	float
- _tilt_angle:	float
- _pan_angle:	float
+ AP_Mount(Location, GPS, AP_AHRS, int):	«constructor»
+ configure_msg(mavlink_message_t):	void
+ control_msg(mavlink_message_t):	void
+ status_msg(mavlink_message_t):	void
+ set_roi_cmd(Location):	void
+ configure_cmd():	void
+ control_cmd():	void
+ update_mount_position():	void
+ update_mount_type():	void
+ debug_output():	void
+ get_mount_type():	MountType
- set_mode(MAV_MOUNT_MODE):	void
- set_retract_angles(float, float, float):	void
- set_neutral_angles(float, float, float):	void
- set_control_angles(float, float, float):	void
- set_GPS_target_location(Location):	void
- calc_GPS_target_angle(Location):	void
- stabilize():	void
- closest_limit(int, int, int):	int
- move_servo(int, int, int, int):	void
- angle_input(RC_Channel, int, int):	int
- angle_input_rad(RC_Channel, int, int):	float

AP_Mount «Modified»	
- _ahrs:	AP_AHRS
- _gps:	GPS
- _current_loc:	Location
- _target_GPS_location:	Location
- _mount_type:	MountType
- _roll_idx:	int
- _tilt_idx:	int
- _pan_idx:	int
- _open_idx:	int
- _roll_control_angle:	float
- _tilt_control_angle:	float
- _pan_control_angle:	float
- _roll_angle:	float
- _tilt_angle:	float
- _pan_angle:	float
+ AP_Mount(Location, GPS, AP_AHRS, int):	«constructor»
+ configure_msg(mavlink_message_t):	void
+ control_msg(mavlink_message_t):	void
+ status_msg(mavlink_message_t):	void
+ set_roi_cmd(Location):	void
+ configure_cmd():	void
+ control_cmd():	void
+ update_mount_position(Location, bool):	void
+ update_mount_type():	void
+ debug_output():	void
+ get_mount_type():	MountType
- set_mode(MAV_MOUNT_MODE):	void
- set_retract_angles(float, float, float):	void
- set_neutral_angles(float, float, float):	void
- set_control_angles(float, float, float):	void
- set_GPS_target_location(Location):	void
- calc_GPS_target_angle(Location):	void
- stabilize():	void
- closest_limit(int, int, int):	int
- move_servo(int, int, int, int):	void
- angle_input(RC_Channel, int, int):	int
- angle_input_rad(RC_Channel, int, int):	float

Figure 11: Original and Modified Class Diagrams for APM Gimbal Mount

Loiter Direction

Early familiarization flights with the APM found that the default ArduPlane firmware (version 2.68 available from community APM repository) [12] only allowed for loitering behavior in a righthand direction (clockwise when viewing from above). This was the case for loiter mode, full auto mode with a loiter waypoint, and guided mode (utilized by follow-me mode).

Although the fixed loiter direction did not preclude the use of follow-me mode, it was clear from the familiarization flights that the aircraft would often make unnecessary

control efforts (in the form of turns greater than 90° in heading change) in order to enter a righthand loiter even if already in a tangential orientation to the desired radius requiring no effort to enter a lefthand loiter. It was decided that allowing the air vehicle to loiter either direction based on real-time conditions would provide the greatest opportunity to match the generated optimal path.

A relative bearing function was introduced to the firmware navigation file, allowing the loiter logic to determine the angle from the current heading of the aircraft to the ground target. This function provided an assessment of how much effort would be associated with entering a loiter in either direction. The loiter could be changed from righthand to lefthand by reversing the sign of the calculated $\Delta_{NavBearing}$ introduced in Chapter 2, Figure 10, based upon the relative distance of the aircraft to the ground target, loiter radius line, and loiter range line. Note that for this research effort, tests executed in the original follow-me mode were intended to baseline unmodified performance (excepting sensor gimbal actuation) so dynamic loiter directions were not activated for phase one flights. They were used for all subsequent tests.

Flight Test Results and Data Analysis

Flight test for the research effort began after all pertinent settings had been identified, experimentation was designed, and requisite firmware modifications were made. Familiarization efforts and tests of initial modifications were all executed in real-world flight tests. For the planned test phases, the range was made available for two separate date ranges.

The primary objective of the first event was collection of all flight data points associated with designed experimentation to determine optimal settings. The loss of the primary aircraft on takeoff and hardware integration problems with backup aircraft resulted in collection of all data points using a simulated Rascal in a HIL environment. The second test event was intended to serve primarily as a demonstration of the final proposed navigation logic, but was limited due to weather. Three flights were executed but only the first, a replicate of basic follow-me mode, was done so within the wind limits of the Rascal airframe. To account for these conditions, all presented data analysis was done on flights executed in a HIL environment. Results from real-world flights are shown for reference, but to ensure consistency, HIL flights are used anywhere a statistical inference is required.

Analysis of Optimality

The objective of experimentation and design for this effort was the minimization of the objective cost function (cost) associated with flights executed in real-time by the autopilot. Analysis necessary to achieve the design work required not only the cost associated with a given flight, but an observation of instantaneous contribution to cost versus time. To measure cost contribution for a discrete point, the derivative of Livermore's proposed cost function [1] was taken and defined in Equation 2 as J_i.

Equation 2

$$J_i = \alpha \left(\frac{SR_i - SR_{desired}}{SR_{desired}} \right)^2 + (1 - \alpha) \left(\frac{u_i}{u_{max}} \right)^2$$

Where:
$i = state\ at\ current\ time$
$\alpha = relative\ weight$
$SR = slant\ range$
$u = roll\ rate$

Note that defining J_i in such a manner allowed each flight to be profiled over time to provide more information than total cost alone. Flights could now be divided into segments of time based on a selected threshold for J_i to determine a relationship between flight conditions and contribution to cost. Charts like the example in Figure 12 were used for analysis and validation of all flight tests.

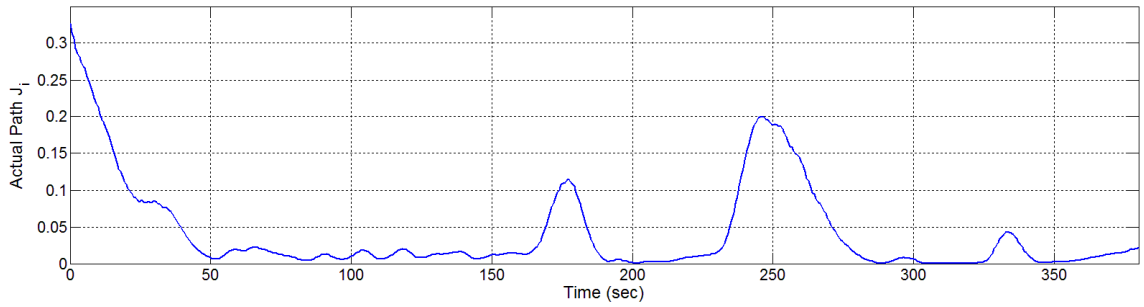

Figure 12: Example Analysis of J_i

Follow-Me

Phase one of flight tests for the research effort was characterization of baseline cost performance for comparison with subsequent design work. An initial flight was conducted for the purpose of flight path analysis and cost profiling. Three additional replicate flights were flown to validate results. Figure 13 shows the flight path with these settings (which results in a cost, *J*, of 9.732) as well as the associated optimal route while Figure 14 depicts the achieved cost profile. These flights were all conducted with a desired slant range of 212m, which equates to a 150m radius when flying at an altitude of 150m.

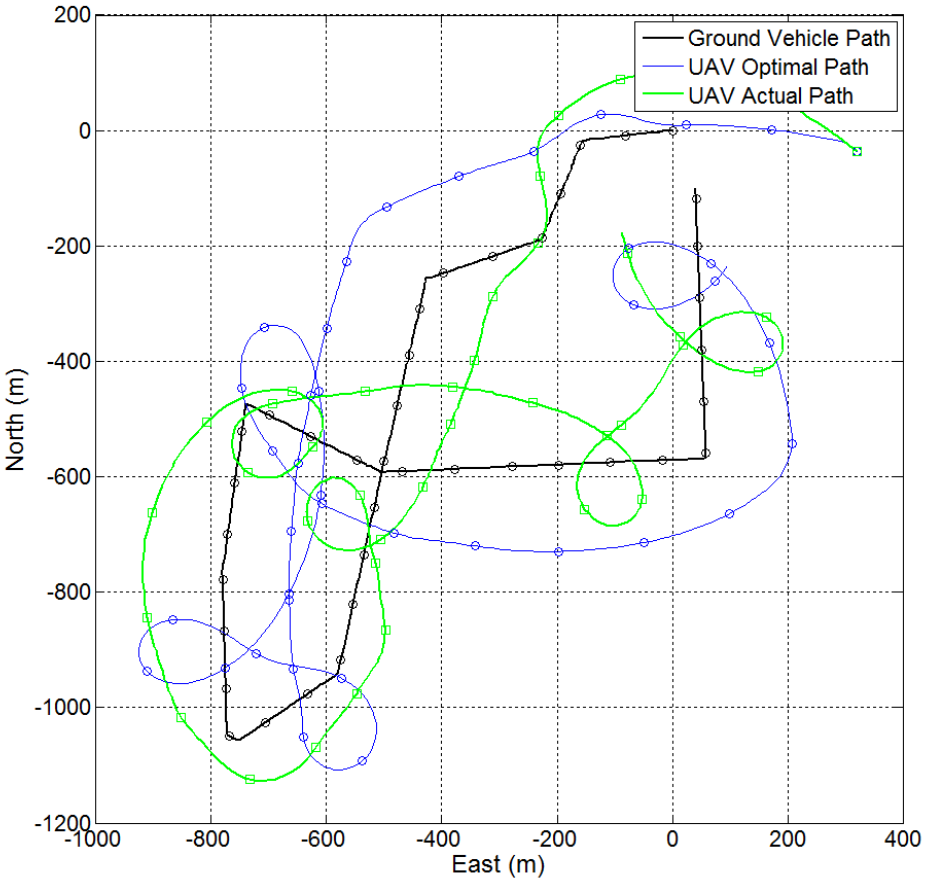

Figure 13: Flight Path with Basic Follow-Me Settings

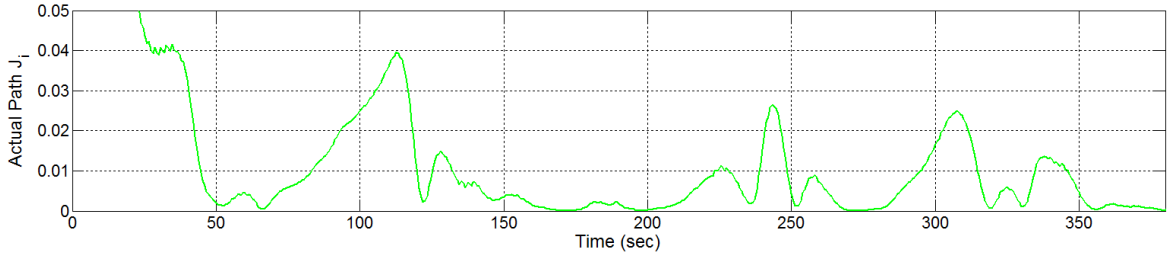

Figure 14: Analysis of J_i for Basic Follow-Me Flight

Optimal Settings Experimentation

After characterizing performance of the unmodified follow-me mode, the first designed experiment was executed. This experiment consisted of 16 flights with loiter

radius, loiter range, and lead time at varied settings coded for analysis in the consequent regression model. Each of the flights was conducted with the air vehicle in starting conditions as similar as achievable by the operator. The simulated ground vehicle drove an identically repeatable preprogrammed course representing the course available on the Camp Atterbury test range. The HIL wind model was stochastic with the average defined as 3.1 m/s (found as real-world average during familiarization flights). Identical settings were used for all flights. Note that the combined starting conditions are used for flights conducted in all three research phases, and are not exclusive to the experimentation portion of the work. Table 1 shows the coded levels for the first stage CCD experiment. High, low, and center values are denoted with a +, -, or 0, respectively. Axial values are denoted with either an "a" or "A." Table 2 summarizes the response results of these flights. Treatment labels are a concatenation of coded levels for loiter range, radius, and lead time in order, with 0 representing all center values.

Table 1: Coded Units for First Stage Flight Experimentation

	Associated Engineering Units		
Coded Level	**Loiter Range (m)**	**Loiter Radius (m)**	**Lead Time (s)**
a	40	50	0
-	58	67	1.1
0	120	125	5
+	182	183	8.9
A	200	200	10

Table 2: Cost Results for First Stage Flight Experimentation

Flight (Test Point)	Treatment	Cost (J, α=.95)
1	+++	4.323
2	a00	9.651
3	--+	25.999
4	00a	55.478
5	0	44.775
6	-++	7.750
7	0	46.970
8	A00	148.387
9	+-+	111.477
10	++-	7.066
11	+--	143.779
12	00A	47.530
13	-+-	9.099
14	0a0	50.445
15	0A0	29.020
16	---	109.754

Once the data was collected, a regression model could be built using the statistical model generated in conjunction with the experimental design. The effects of experimental factors (radius, range, and lead time) in the model were found to be significant (p-value < .05) as presented in Table 3. The model terms are presented in Table 4 sorted in order of estimate magnitude.

Table 3: Analysis of Variance for First Stage Flight Experimentation

Source	Degrees of Freedom	Sum of Squares	Mean Square	F Statistic
Model	5	26057.933	5211.59	5.3697
Error	10	9705.456	970.55	**Prob > F**
Total	15	35763.390		0.0118

Table 4: Sorted Parameter Estimates for First Stage Flight Experimentation

Term	Estimate	Std Error	t Ratio	Prob > |t|
Loiter Radius	-34.5023	9.262	-3.73	0.0039
Loiter Range	25.8644	9.262	2.79	0.019
Loiter Range * Loiter Radius	-15.6204	11.0145	-1.42	0.1865
Loiter Radius * Lead Time	13.9956	11.0145	1.27	0.2326
Lead Time	-11.524	9.262	-1.24	0.2418

Note that the experiment indicated only loiter radius and loiter range to be statistically significant at the $\alpha = .05$ level. A factor profile, shown in Figure 15, was generated based on the model to help select the recommended settings. The resultant recommendations were to set loiter radius to its highest setting and range to its lowest, which for this experiment translated to a radius of 200m and a range of 40m. It was decided to select 150m as the recommendation for radius, based on the real-world safety requirement that the pilot must maintain visual contact with the SUAS. Lead time was suggested to be set at zero, even though it was not significant.

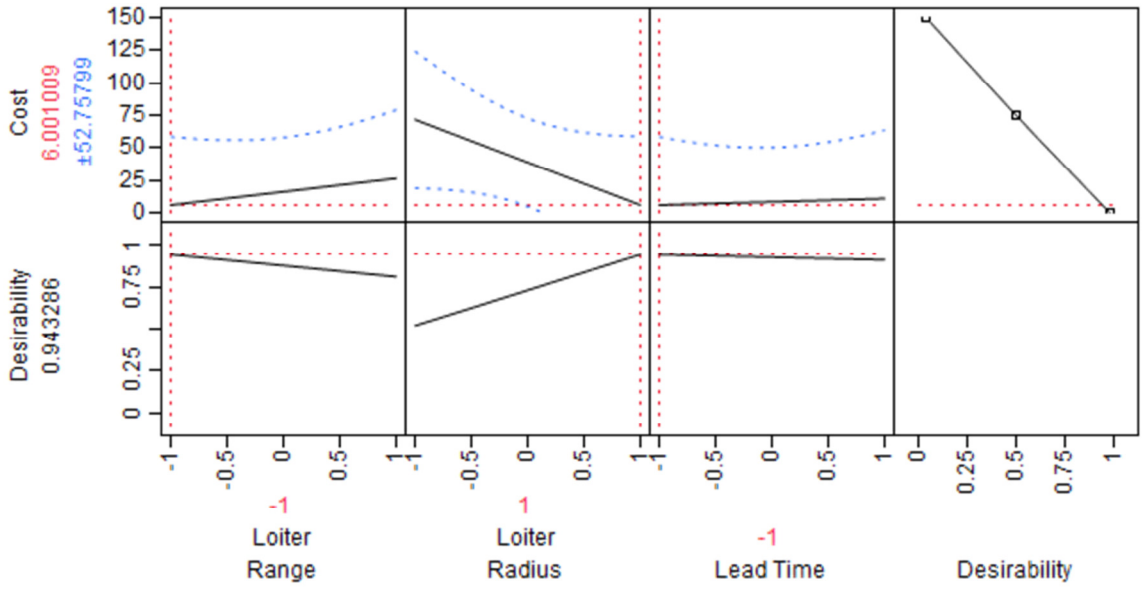

Figure 15: Factor Profiler for First Stage Experimental Model

To validate the results and increase confidence in recommended settings, a second stage experiment with finer granularity was designed around the 150m radius and 40m range test space. Due to the lack of significance, lead time was excluded from this experiment in all but one center point replicate and set to zero for all flights. Coded units for the CCD are shown in Table 5 and cost results after completion are shown in Table 6.

Table 5: Coded Units for Second Stage Flight Experimentation

	Associated Engineering Units	
Coded Level	**Loiter Range (m)**	**Loiter Radius (m)**
a	18	108
-	20	110
0	40	130
+	60	150
A	62	152

Table 6: Cost Results for Second Stage Flight Experimentation

Flight (Test Point)	Treatment	Cost (J, α=.95)
1	0A	19.288
2	-+	17.185
3	A0	25.146
4	0	30.076
5	0	21.442
6	--	27.742
7	+-	22.606
8	a0	31.880
9	0a	21.574
10	++	23.250

Once flights were conducted, a second regression model was built using the generated model on which the experimental design was based. Analysis of second model, shown in Table 7, finds that the included terms do have a statistically significant impact (p-value < .05) on the cost of a flight.

Table 7: ANOVA for Second Stage Flight Experimentation

Source	Degrees of Freedom	Sum of Squares	Mean Square	F Statistic
Model	4	167.586	41.90	7.3047
Error	4	22.942	5.74	**Prob > F**
Total	8	190.528		0.0400

Parameter estimates for this model, shown sorted in Table 8, only indicated the significance of $loiter\ radius^2$ at the $\alpha = .05$ level, resulting in the quadratic profile seen in Figure 16. The profiler, in agreement with the first stage experiment, suggested a high setting for the loiter radius. Due to safety limitations, the highest recommendation for radius remained 150m. Loiter range, although not statistically significant, still showed a negative regression parameter estimate in agreement with the first stage model. The lowest non-axial loiter range treatment used in the second test, 20m, became the recommended setting.

Table 8: Sorted Parameter Estimates for Second Stage Flight Experimentation

Term	Estimate	Std Error	t Ratio	Prob > |t|
Loiter Radius * Loiter Radius	-6.8416	1.594	-4.29	0.0127
Loiter Range * Loiter Radius	2.8003	1.197	2.34	0.0795
Loiter Radius	-1.9571	0.952	-2.06	0.1091
Loiter Range	-1.0010	0.952	-1.05	0.3525

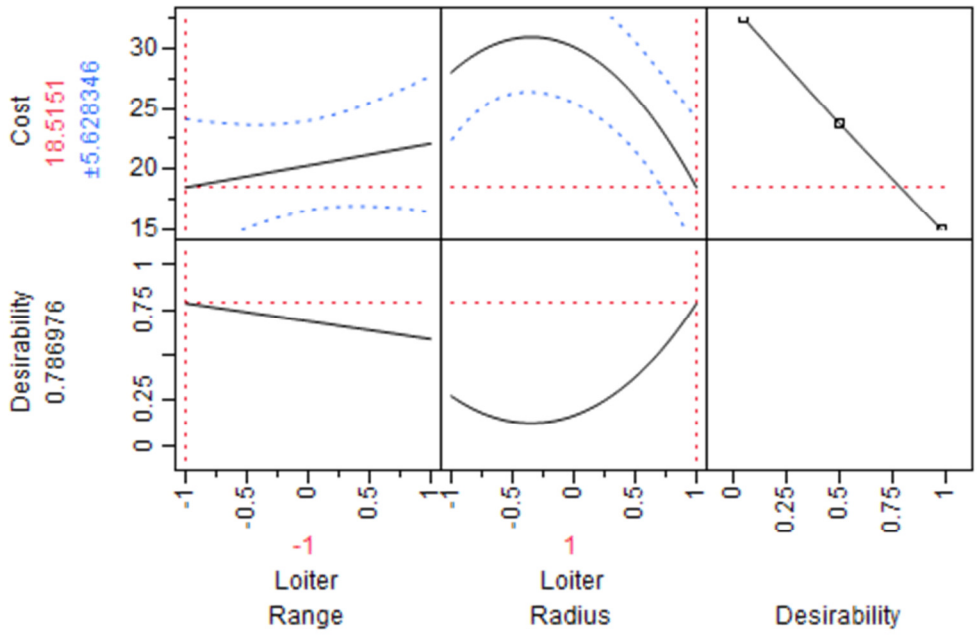

Figure 16: Factor Profiler for Second Stage Experimental Model

After both stages of experimentation were complete, demonstration flights were done at the final recommended settings of 150m loiter radius, 20m loiter range, and no lead time. One initial flight was done for analysis purposes, with two additional replicates for validation. These were treated separately from the test point at these settings flown during experimentation, which served as a third replicate at the suggested settings. Figure 17 shows the demonstration flight path ($J = 15.185$) as well as the associated optimal route while Figure 18 depicts the achieved cost profile. Note that optimal flight paths are calculated based on real wind data telemetry from each associated test. The result is that even though starting conditions were common for all flights in the research effort, calculated optimal paths are not all identical.

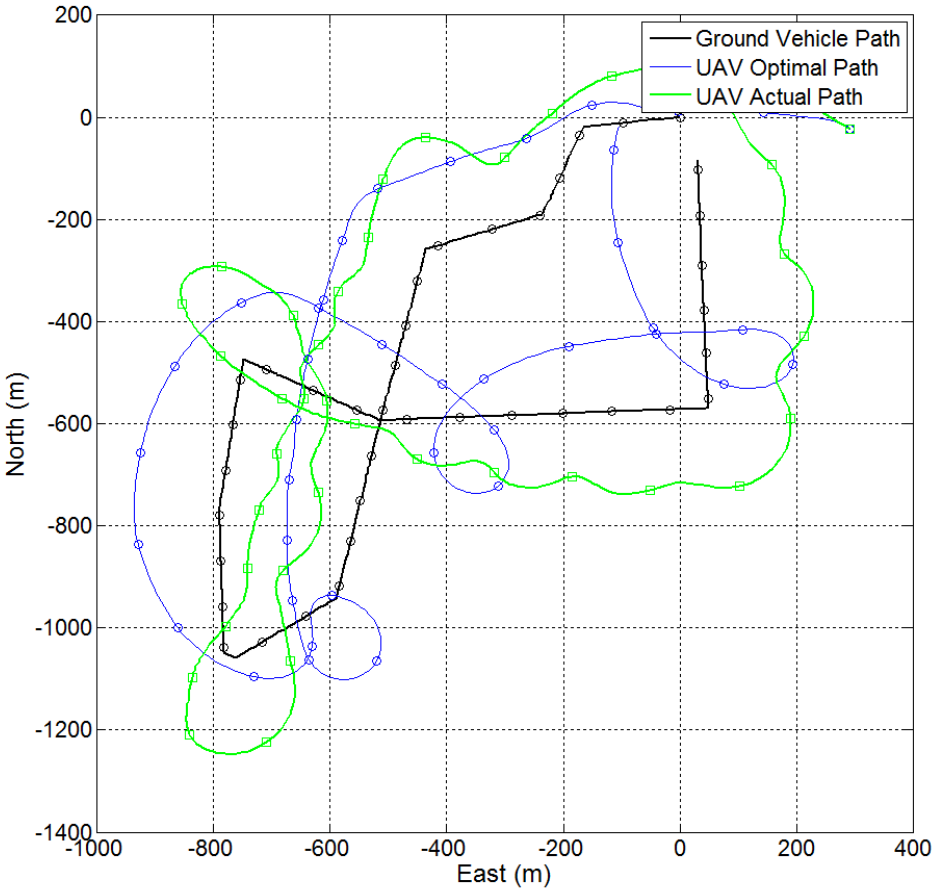

Figure 17: Flight Path Using Settings Determined by Experimentation

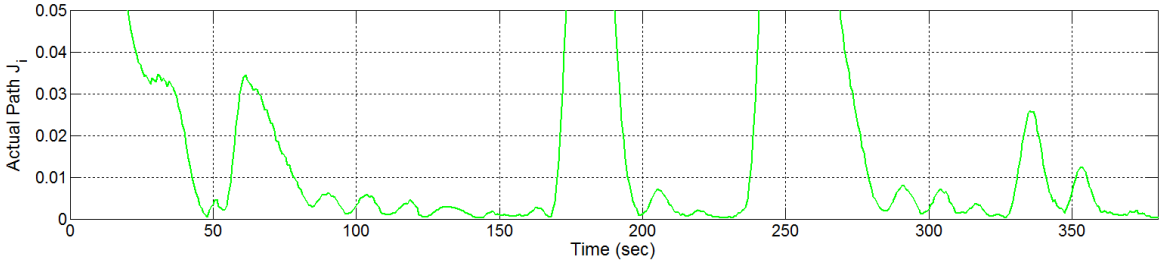

Figure 18: Analysis of J_i for Settings Determined by Experimentation

Finite State Machine First Iteration

After completing experimentation, arriving at recommended settings, and profiling associate cost performance, design was done on a heuristic approach to further improve performance. It is important to note that the ArduPlane firmware is written as an Arduino sketch, using a combination of C++ libraries and traditional Arduino code for main processes. Arduino sketches are run as loops, using conditional statements to vary behavior and timing. Therefore, the natural way to implement heuristic logic is to assess the system state iteratively and execute the desired reaction using switch conditions, which allows cases to be defined and run selectively. When examining any single process loop, the implementation of mutually exclusive selective cases is the equivalent of a finite state machine (FSM).

To design such a state machine for the purpose of minimizing cost during convoy overwatch missions, states were defined in which alternate behavior is required. In Figure 19, the J_i profile for the demonstration of experimentally suggested settings was plotted over the turn rate of the ground vehicle being driven. It was found that both of the time segments with large increases in J_i are immediately preceded by substantial turns made by the ground vehicle.

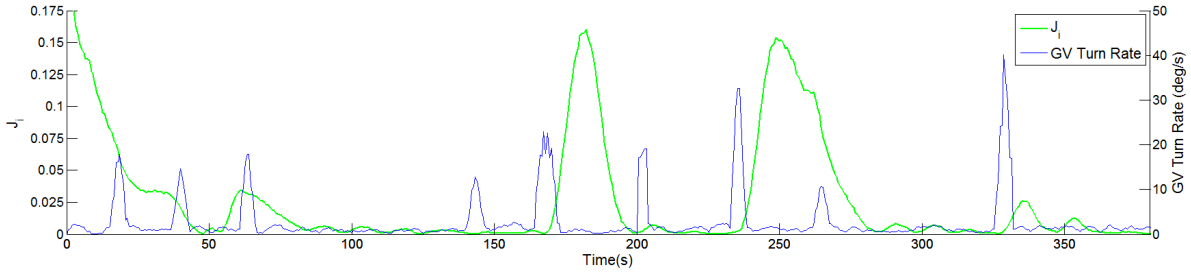

Figure 19: J_i Compared Against Ground Vehicle Turn Rate

Also of note is that while both large increases in J_i followed periods of high ground vehicle turn rate, not all ground vehicle turns result in J_i growth. Figure 20 shows the air and ground vehicle paths highlighting the period of time encapsulating the second large peak in J_i, from time = 235-280s. It was found that both periods of increased cost correspond to a common situation in which the ground vehicle turned such that it was heading in a divergent direction from the air vehicle. Even when the aircraft commanded a full effort turn, the time required to return to the desired slant range resulted in large cost contributions if these two headings were initially opposite.

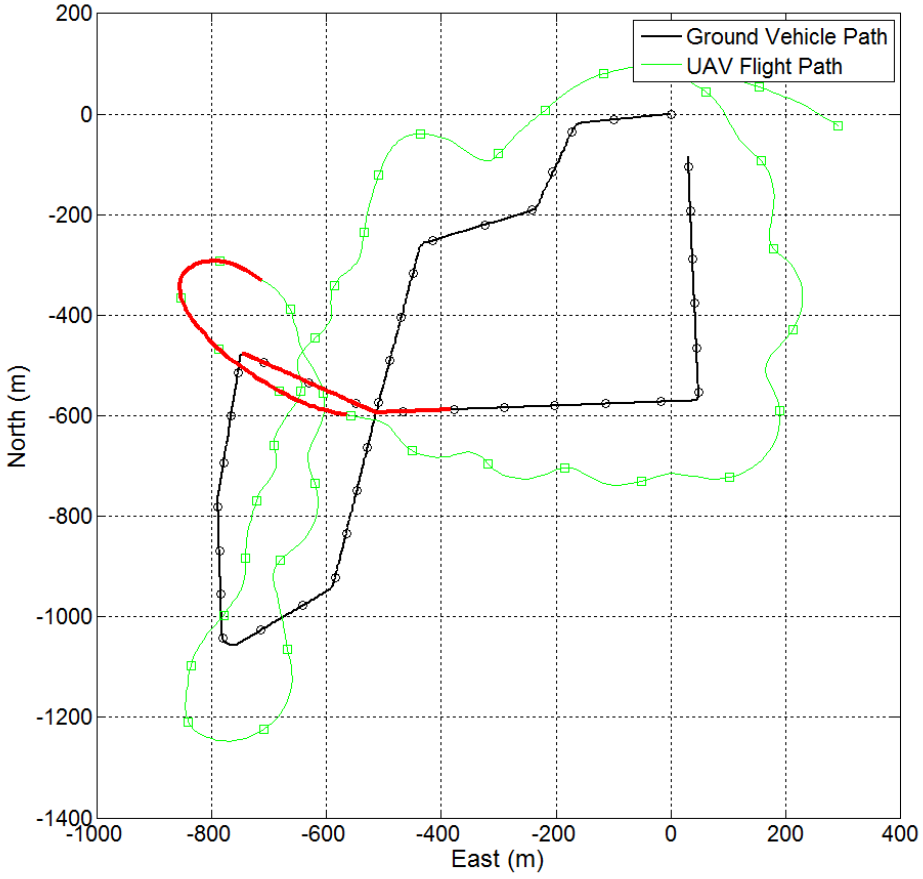

Figure 20: Highlighted Portion of Flight Test with Increased J_i

To account for this scenario, an FSM was constructed which allowed for tighter turns in the event that the air vehicle was both conducting a full effort turn and J_i increased past a given threshold. In this additional state, a multiplier was used to temporarily decrease the output throttle setting, causing a reduced turn radius. The designed FSM is diagrammed in Figure 21.

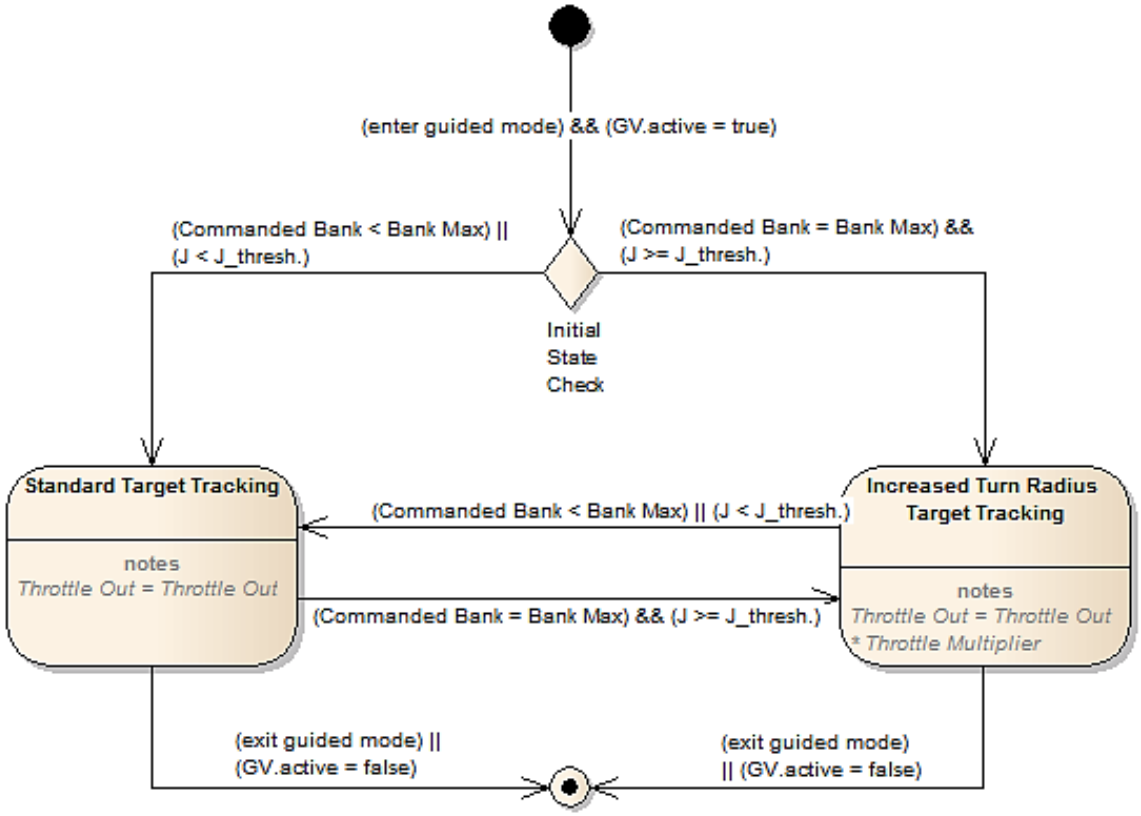

Figure 21: Finite State Machine Initial Design

The initial demonstration of the FSM was flown with $J_{threshold}$ set to 0.04 and the throttle multiplier at 0.75. $J_{threshold}$ was selected based on the evaluated J_i profile in an attempt to detect true peaks and avoid unnecessary state transition based on minor oscillation. The throttle multiplier was selected as a conservative value meant to

noticeably decrease the associated turn radii without causing excessive control behavior or risking stalled conditions. The flight path for the initial demonstration, as well as the calculated optimal path, is shown in Figure 22, achieving a cost of 5.110. The accompanying J_i profile is shown in Figure 23. In addition, two replicate flights were executed.

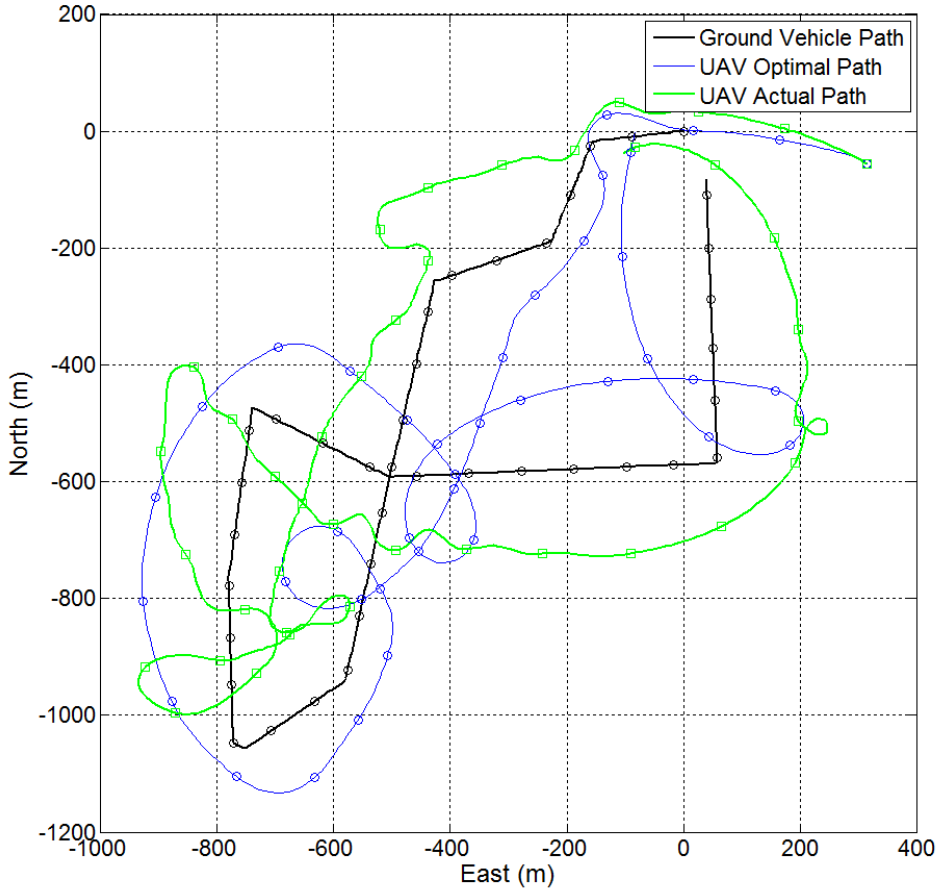

Figure 22: Flight Path Using Initial State Machine Logic

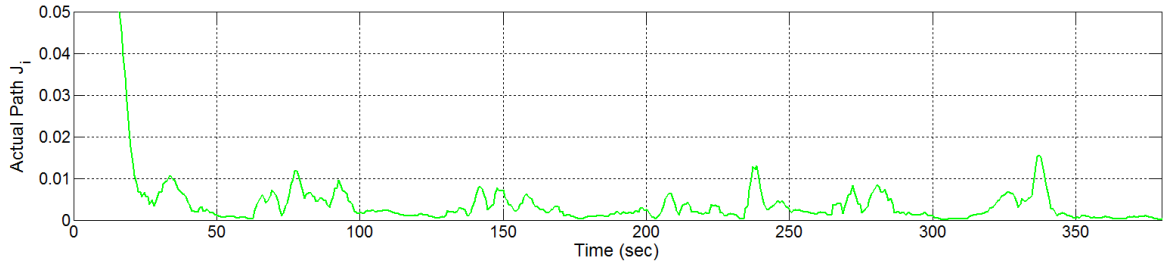

Figure 23: Analysis of J_i for Initial State Machine Logic

Experiment Review and Finite State Machine Second Iteration

After completion of all three flight test phases (follow-me, experimentation, and FSM design), including replicates, cost data was combined and compared. Table 9 shows these results, to include averages. Note that real-world tests of each mode were included for reference, but not included in statistical analysis.

Table 9: Summary of Cost Results from Initial Tests and Follow-On Replicates

		Basic Follow-Me	NonHeuristic Optimal Settings	Finite State Machine
Replicate	**Environment**	**Cost (J, α=.95)**	**Cost (J, α=.95)**	**Cost (J, α=.95)**
Initial Test	HIL	9.732	15.185	5.110
Rep 1	HIL	6.747	4.964	3.100
Rep 2	HIL	5.656	3.325	3.598
Rep 3	HIL	4.915	N/A	N/A
CCD Test Point	HIL	N/A	17.185	N/A
Flight Test	Real World	9.171	6.701	8.285
Average		7.244	9.472	5.023
Average (HIL Only)		**6.762**	**10.165**	**3.936**

The most noteworthy observation from these results is that the settings determined to be most optimal through experimentation in fact achieved worse performance than basic follow-me settings. In addition, flights at these settings had a much wider cost variance than either follow-me or FSM, signifying inconsistent performance. This inconsistency, coupled with degraded average performance compared

to follow-me, indicates that analysis of initial experimental data failed to properly characterize key effects.

The data from flight experimentation was reassessed by combining all 26 original test points in non-coded form (engineering units) and creating a traditional regression model. Note that lead time was zero for all second stage flights. Terms were considered to the three factor interaction level and screened before inclusion in the model. Table 10 shows the screener results with considered terms, based on contrast, highlighted. This was validated by Figure 24, a half normal plot indicating potential term significance.

Table 10: Screener for Factor Inclusion in Combined Data Regression Model

Term	Contrast	t-Ratio	Individual p-Value
Radius	-23.0789	-7.24	0.0002
Range	20.7977	6.52	0.0003
Lead Time	-7.2107	-2.26	0.0368
Radius * Radius	-3.0166	-0.95	0.3333
Radius * Range	-9.6744	-3.03	0.0121
Range * Range	5.7567	1.81	0.0825
Radius * Lead Time	7.0742	2.22	0.0394
Range * Lead Time	-0.8153	-0.26	0.8067
Lead Time * Lead Time	-3.5546	-1.11	0.2591
Radius * Radius * Radius	6.7533	2.12	0.0472
Radius * Radius * Range	-13.3488	-4.19	0.0031
Radius * Range * Range	-6.4166	-2.01	0.0563
Range * Range * Range	-2.0162	-0.63	0.5529
Radius * Radius * Lead Time	-4.067	-1.28	0.2009
Radius * Range * Lead Time	-1.4136	-0.44	0.6699
Range * Range * Lead Time	0.2447	0.08	0.9440
Radius * Lead Time * Lead Time	-8.105	-2.54	0.0235
Range * Lead Time * Lead Time	1.3055	0.41	0.6965
Lead Time * Lead Time * Lead Time	1.8446	0.58	0.5856

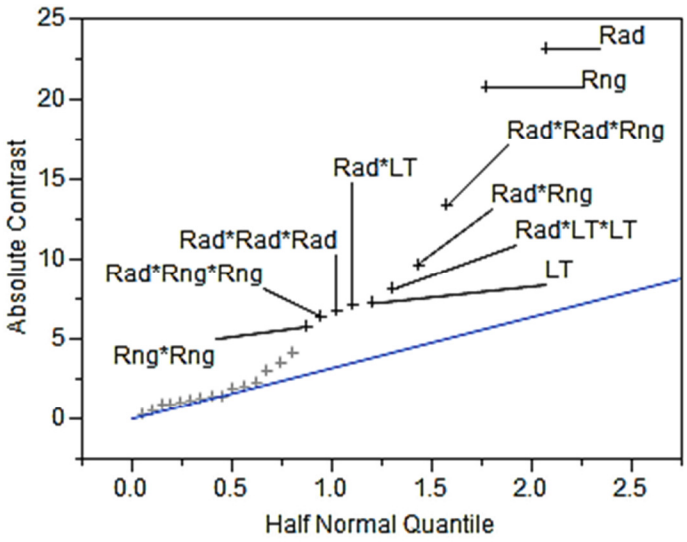

Figure 24: Half Normal Plot with Significant Terms Labeled

The terms proposed in the screener were used to construct a new, more complex, regression model with a value 0.931 for $R^2_{adjusted}$. Table 11 validates that the models effect on cost was significant and the final model estimates are shown in Table 12.

Table 11: ANOVA for Combined Data Set with Selected Factors

Source	Degrees of Freedom	Sum of Squares	Mean Square	F Statistic
Model	10	39498.222	3949.82	34.6455
Error	15	1710.102	114.01	**Prob > F**
Total	25	41208.325		<.0001

Table 12: Sorted Parameter Estimates for Combined Regression Model

Term	Estimate	Std Error	t Ratio	Prob > |t|
Range	0.591026	0.064623	9.15	<.0001
(Radius-126.923)*(Radius-126.923)*(Range-89.231)	-0.000189	0.000026	-7.19	<.0001
(Range-89.231)*(Range-89.231)	0.004299	0.000859	5.01	0.0002
(Radius-126.923)*(Radius-126.923)*(Radius-126.923)	-0.000617	0.000149	-4.14	0.0009
(Radius-126.923)*(LeadTime-3.269)	0.455661	0.112129	4.06	0.0010
LeadTime	-2.641159	0.690165	-3.83	0.0017
Radius	2.944517	0.773584	3.81	0.0017
(Radius-126.923)*(LeadTime-3.269)*(LeadTime-3.269)	-0.114114	0.031304	-3.65	0.0024
(Radius-126.923)*(Range-89.231)*(Range-89.231)	-0.000072	0.000080	-0.90	0.3808
(Radius-126.923)*(Range-89.231)	-0.000718	0.004953	-0.14	0.8867

A factor profiler, shown in Figure 25, was constructed for the new model to help graphically determine the best combination of settings. The results of the new model were in fact different from the first iteration of experimental data analysis. The recommended settings from the combined regression analysis were a 100m loiter radius, a 65m loiter range, and a 3s lead time. In this case, all three were determined to be significant at the $\alpha = .05$ level.

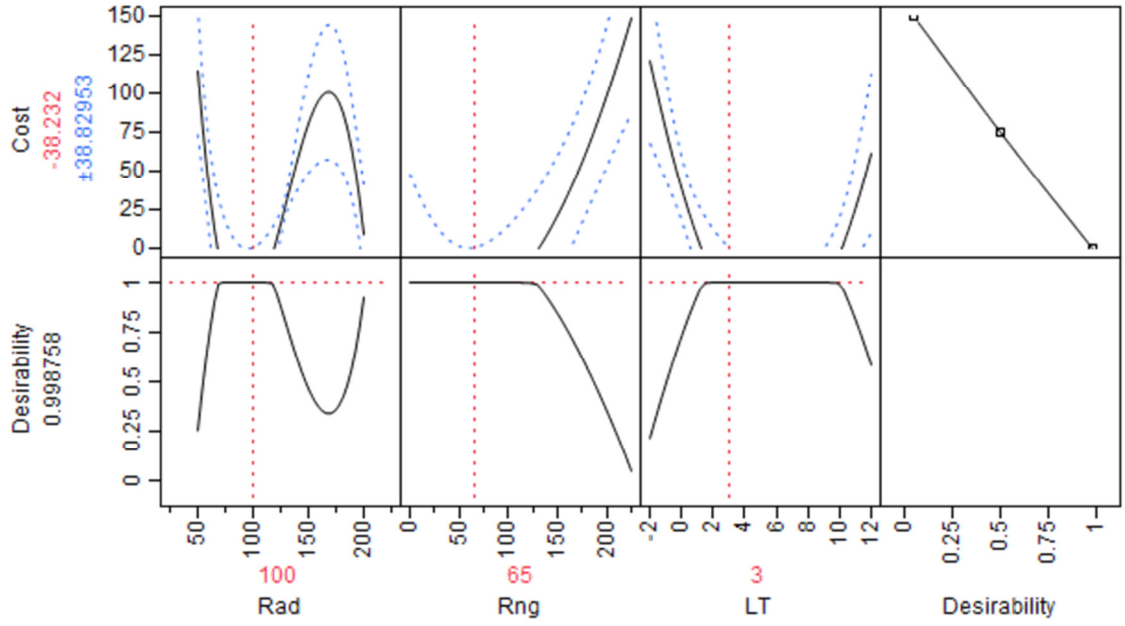

Figure 25: Factor Profiler for Combined Regression Model

Demonstration flights were conducted with the lead time reintroduced at 3s, loiter range increased to 65m, proposed FSM functionality disabled, and loiter radius left at 150m. Note that the suggested setting of 100m was not used to allow for comparison with results from existing tests. The flight path achieved a cost of $J = 2.799$ and is shown in Figure 26. The associated J_i profile is shown in Figure 27. Three additional replicates were flown for validation.

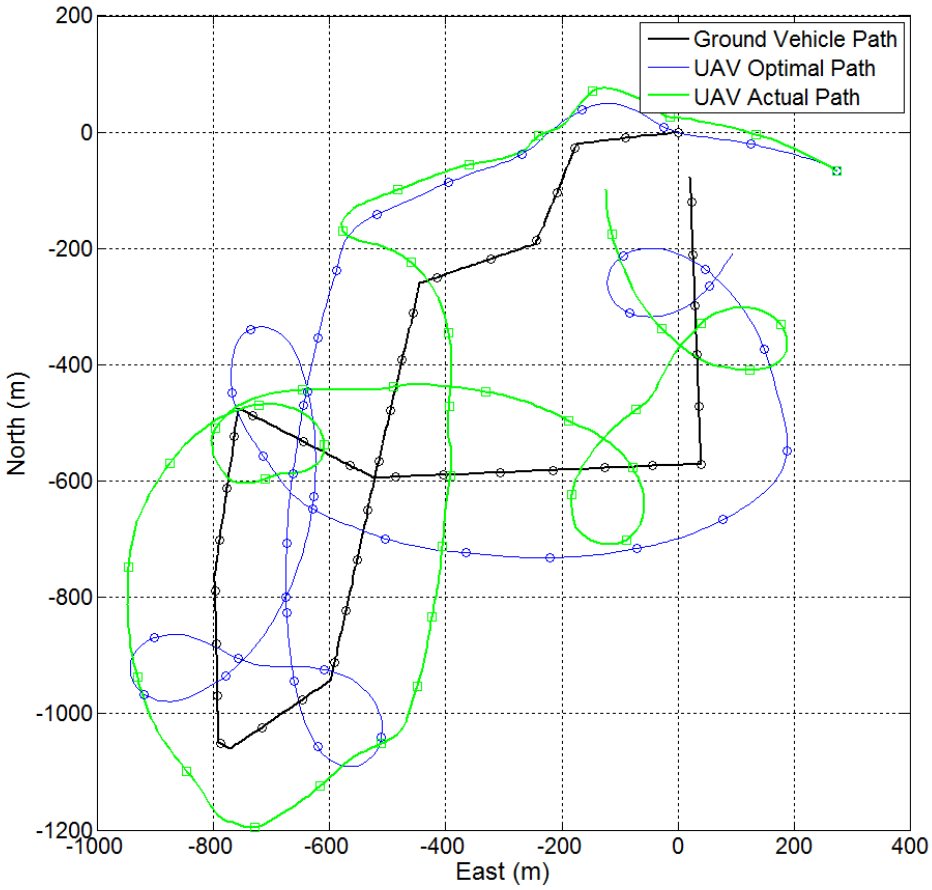

Figure 26: Flight Path for Optimal Settings from Combined Regression Model

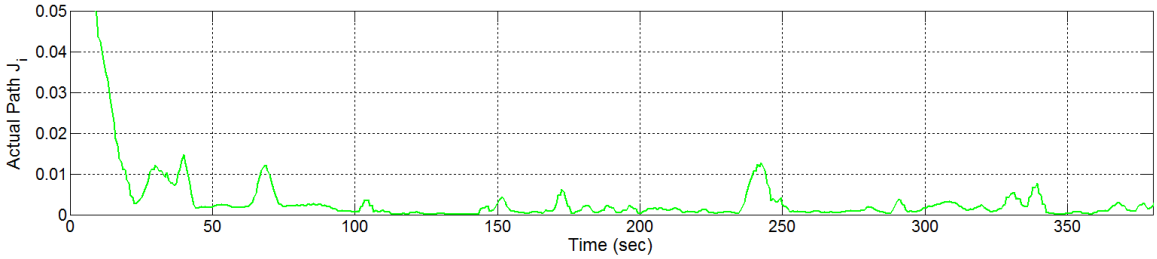

Figure 27: Analysis of J_i for Suggested Settings from Combined Regression Model

Following flight test of resultant settings from the second experimental analysis, a second iteration of FSM design was proposed. Definition of states requiring alternate behavior was not as intuitive as the first design due to an overall increase in performance

with most peaks in J_i not exceeding 0.01. Figure 28 depicts an analysis of slant range for the demonstration flight where it is seen that most increases in J_i correlated to periods during which the aircraft spent inordinate amounts of time off of the desired slant range.

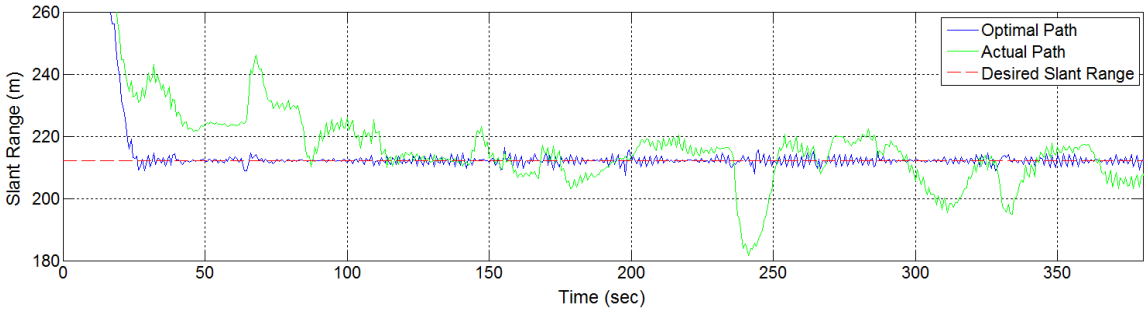

Figure 28: Slant Range Analysis Generated for Suggested Settings Flight Path

This relationship was expected given that α, as defined in Equation 1, was set to 0.95 for all of Livermore's optimization functions used in this effort [1], heavily favoring slant range. However, when compared to the actual flight path, the analysis helps demonstrate that flight times with poor slant range performance are typically those in which the SUAS overcame the ground vehicle while both were traveling in relatively straight paths with common headings. Under these circumstances, it was found that the air vehicle occasionally found itself unable to commit to either a full turnaround or increased effort to regain the desired slant range.

A second iteration FSM was proposed that, when appropriate, attempted to diminish the effects of this scenario by scaling the level of effort being used to maintain slant range. This design, with all associated transition logic is diagramed in Figure 29.

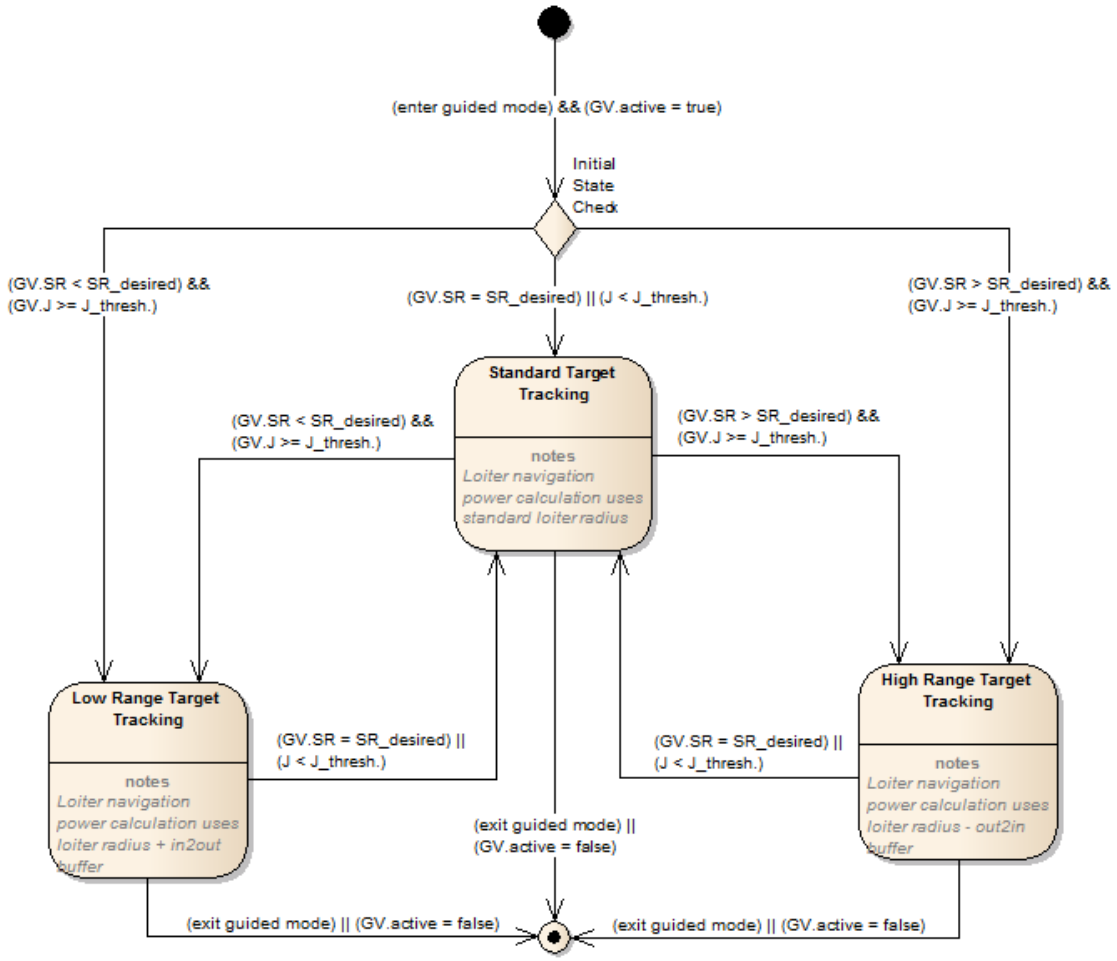

Figure 29: Revised Finite State Machine

The second iteration FSM was implemented on the APM with $J_{threshold} = 0.003$ and control effort buffer set to 35m, representing a ±23% change over a desired radius of 150m. Like the initial FSM, $J_{threshold}$ was selected based on the J_i profile in an attempt to execute state transitions when necessary but not excessively. The control effort buffer was set to a conservative value intended to effect measurable changes without causing unsafe behavior if flown in real-world test. A demonstration flight was conducted,

followed by three additional replicates. The demonstration flight path is shown in Figure 30, with cost of 2.35. The J_i profile for the flight is shown in Figure 31.

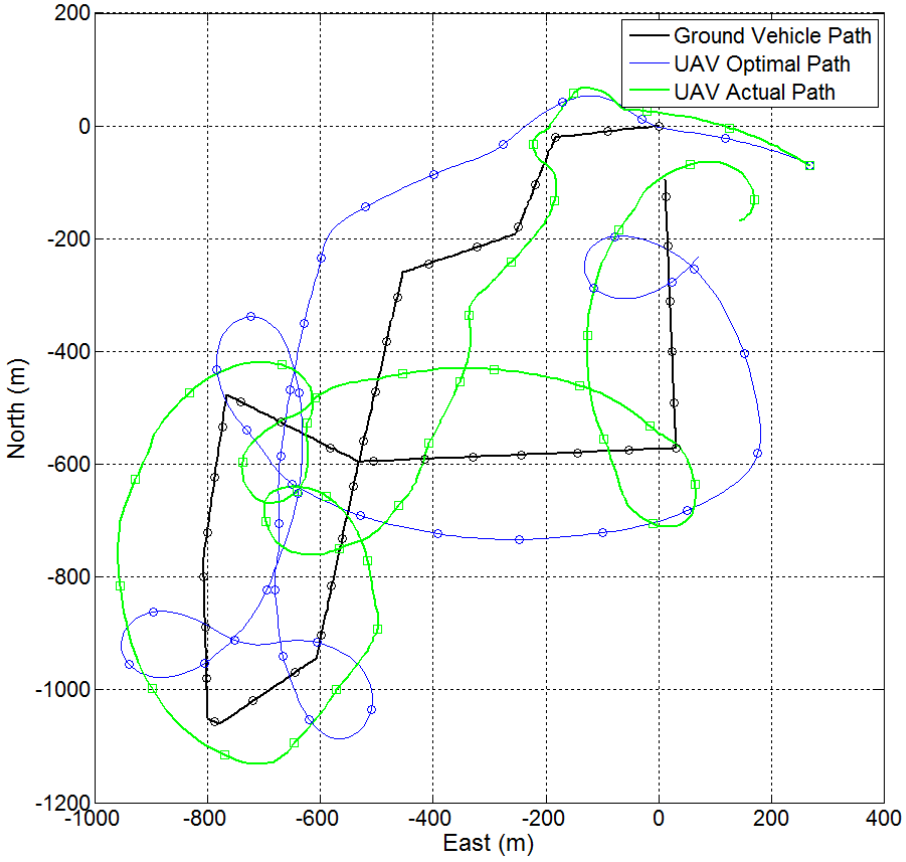

Figure 30: Flight Path for Revised Finite State Machine

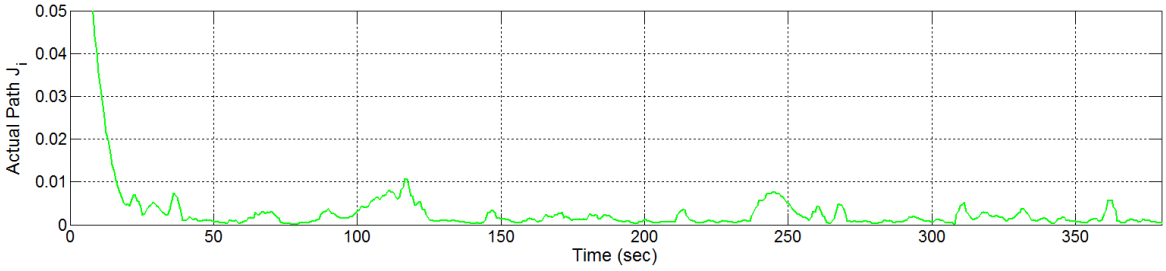

Figure 31: Analysis of J_i for Revised Finite State Machine

For the final replicate flight utilizing the revised FSM design, a debugger was added to the firmware allowing for analysis of which states were active over the course of the flight. This is profiled in Figure 32.

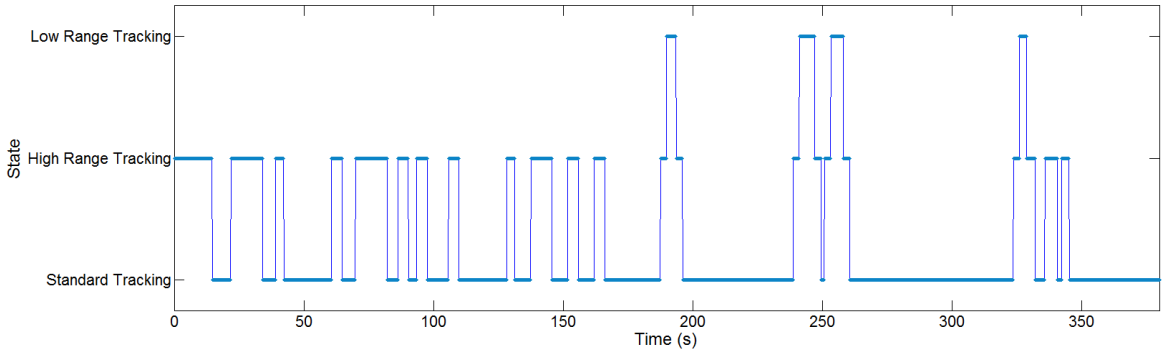

Figure 32: Profile of Current State for Final FSM Flight

While the profile shows that the best setting for state transition conditions may require further experimentation, it does validate that all three states were entered at various points throughout the course of flight. The fact that the majority of the time was spent in the standard tracking state does indicate that both alternate states were effective in their goals of returning the SUAS to a condition with low J_i and low slant range error.

Comparative Results and Investigative Questions

Once all replicate flights were executed with telemetry data appropriately recorded, comparative analyses were conducted both to measure improvement in cost and validate applicability to the convoy overwatch scenario. Table 13 presents a summary of cost results for initial demonstration flights and replicates flown in all three stages of the research effort. For this analysis, note that only the second iteration of experimentally suggested settings and FSM design were considered.

Table 13: Summary of Cost Results after Secondary Data Analysis and State Machine Design

		Basic Follow-Me	DOE Suggested Settings - V2	Finite State Machine - V2
Replicate	**Environment**	**Cost (J, α=.95)**	**Cost (J, α=.95)**	**Cost (J, α=.95)**
Initial Test	HIL	9.732	2.699	2.350
Rep 1	HIL	6.747	5.249	1.966
Rep 2	HIL	5.656	2.799	2.513
Rep 3	HIL	4.915	1.985	2.361
Average		**6.762**	**3.183**	**2.298**
% Improvement over Follow-Me			**52.9%**	**66.0%**

These sets of results were specifically intended to answer the first three investigative questions listed for this effort, restated below:

- What is the target tracking and flight path performance of the SUAS when using a basic follow-me mode?
- What is the best path performance achievable by the adjustment of existing or readily accessible navigation control without implementation of state responsive logic?
- What is the achievable SUAS flight path optimality using a state-based, heuristic approximation of the optimization strategy?
- What is the feasibility of implementing heuristic ground target tracking logic that is capable of real-time execution onboard a SUAS autopilot?

Basic follow-me flights, flights at the experimentally suggested settings, and the proposed FSM flights were all conducted in direct response to first three investigative questions, respectively. A visual depiction of the achieved performance differences is shown in Figure 33, in which the J_i profile for all three initial flight demonstrations are overlapped along with associated average J_i for each.

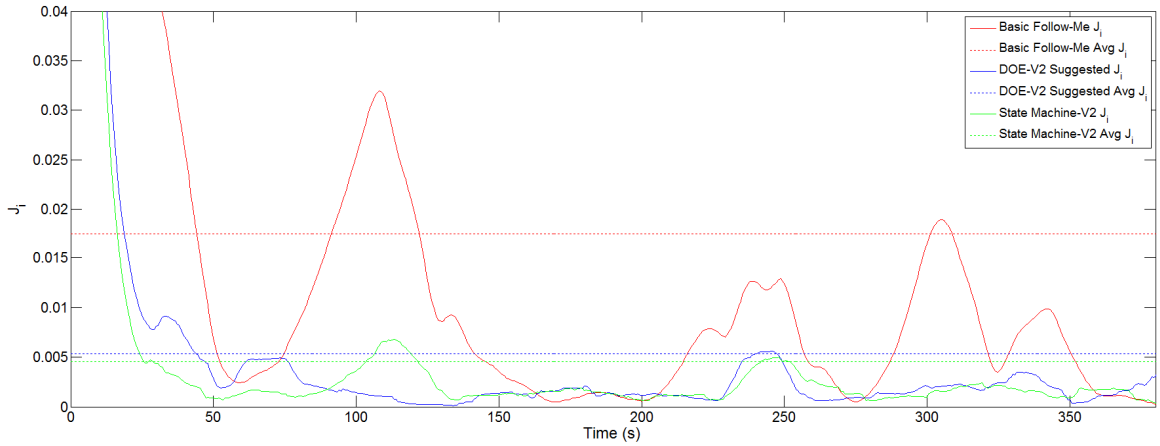

Figure 33: Cost Performance for Initial Flight Tests of Main Configurations

The final investigative question was more complex. From a technical standpoint, two items were required to subjectively assess the viability of heuristic approximation of optimal control: first was a measure of performance increase significance and second was validation that the proposed strategy remains capable of meeting mission requirements.

Basic costs, both individuals and averages, are presented in Table 13, above. However, this does not provide an indication of the significance of achieved results. In order to claim that performance increases can truly be expected from the presented settings and heuristic design, confidence intervals based on the collected samples were compared. Table 14 shows 95% confidence intervals calculated for the true mean performance expected at each of the three demonstrated firmware configurations. The samples used to calculate these intervals were the initial flights and replicates collected at each stage of the effort. For all three configurations n is equal to four replications.

Table 14: 95% Confidence Intervals for Cost Performance of Main Configurations

Firmware Configuration	Achieved Cost (J) Sample Average	$-t_{0.05,3}*(s/n^{0.5})$	$+ t_{0.05,3}*(s/n^{0.5})$
Basic Follow-Me	6.762	3.393	10.132
Final Experimental Suggested Setting	3.183	0.917	5.449
Final FSM Design	2.298	1.926	2.669

Figure 34 depicts the same confidence intervals graphically. This figure is important because it demonstrates that there was no overlap between the basic follow-me and final FSM configurations. The lack of overlap means that the true average performance was in fact been improved over the basic follow-me performance. The same could not be said for the experimentally suggested settings. However, theses settings were intended primarily to provide the requisite analysis for arriving at the final FSM.

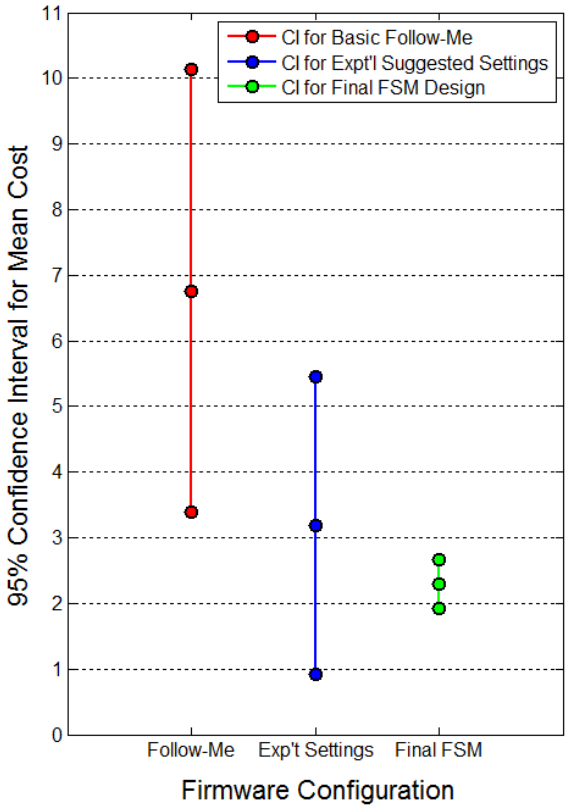

Figure 34: Confidence Intervals for Cost Performance of Main Configurations

Next, flights conducted with the final FSM could be compared to their respective optimal paths (theoretically calculated in MATLAB) to determine differences in performance. Table 15 shows data from all four flights using the proposed firmware with each associated optimal cost, including a 95% confidence interval conducted on each set of four J values. These confidence intervals are depicted graphically in Figure 35.

Table 15: 95% Confidence Interval for Final FSM and Associated Optimal Costs

Replicate	Final FSM Achieved Cost	Associated Optimal Path Cost
Initial Test	2.350	1.449
Rep 1	1.966	1.842
Rep 2	2.513	1.329
Rep 3	2.361	0.838
Sample Average	2.298	1.364
$+ t_{0.05,3}*(s/n^{0.5})$	2.669	2.023
$-t_{0.05,3}*(s/n^{0.5})$	1.926	0.706

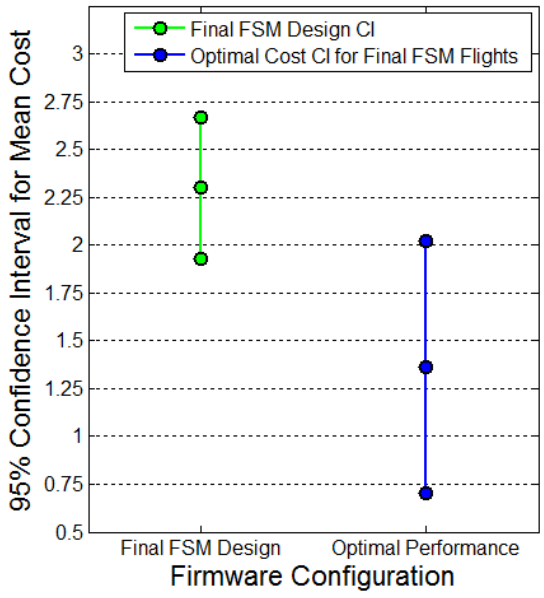

Figure 35: Plotted Confidence Intervals for Final FSM Design and Respective Optimal Paths

Note that using the four replicates conducted, overlapping confidence intervals were found for the achieved and optimal costs. While this was a good indicator that achieved performance was close to the optimal, the overlap was relatively narrow so a more conservative hypothesis test was conducted. A one tailed t-test was used based on the sample sizes and the assumption that the FSM could not perform better than the optimal. The results of this test are shown in Table 16.

Table 16: Two Sample t-Test (Unequal Variance) for Final FSM Flights and Associated Optimal Paths

	Final FSM Achieved Cost	*Associated Optimal Path Cost*
Mean	2.2975	1.3643
Variance	0.054362087	0.171317413
Observations	4	4
Hypothesized Mean Difference	0	
df	5	
t Stat	3.928788685	
P(T<=t) one-tail	0.005541883	
t Critical one-tail	2.015048373	

With $t_{stat} > t_{critical}$, this test rejected the hypothesized difference of zero and indicated that the final FSM did in fact perform worse than the optimal at the $\alpha = 0.05$ level. This was expected as the calculated optimal is based on perfect future knowledge of the ground vehicle path and the proposed FSM is a real-time heuristic making no cost assessments of predicted scenarios.

Finally, to validate that the proposed solution was capable of meeting convoy overwatch mission requirements, sensor time-on-target was evaluated for the follow-me, DOE suggested, and final FSM settings. The HIL environment allowed a virtual sensor gimbal to be added to the SUAS, enabling theoretical time-on-target to be evaluated in an

identical fashion to real world flights. Figure 36 depicts sensor aimpoint for all three respective demonstration flights while Table 17 provides associated percentages for both the actual HackHD lens (160° field of view) and an optional 16.9° lens.

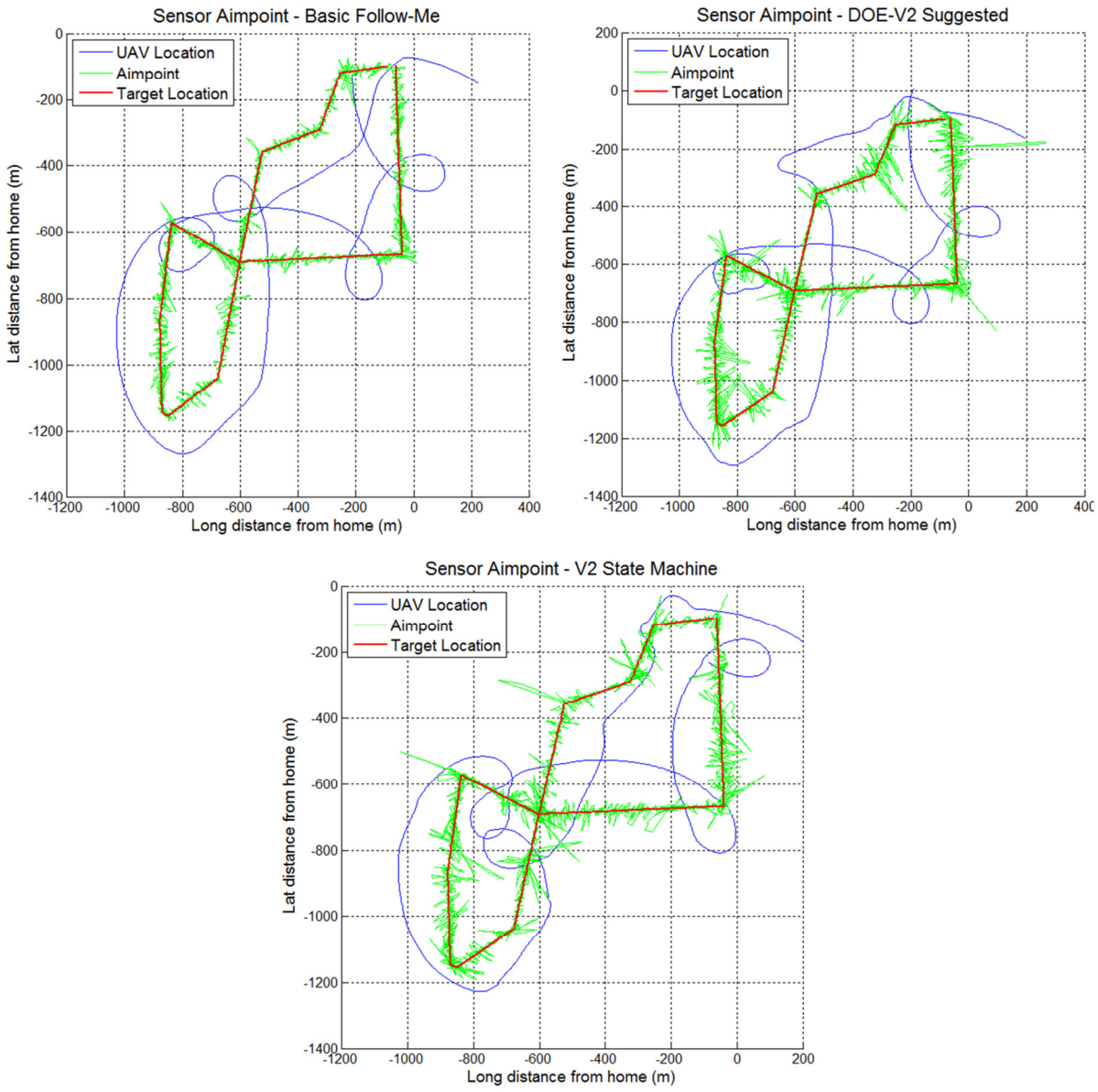

Figure 36: Sensor Aimpoint for Initial Flight Tests of Main Configurations

Table 17: Sensor Time-on-Target Performance for Initial Flight Tests of Main Configurations

Flight	Percent Time-on-Target with Stock 160° Lens	Percent Time-on-Target with Optional 25mm 16.9° Lens
Basic Follow-Me	100%	100%
DOE Suggested Settings V2	99%	92%
FSM Final Design	100%	92%

Findings show that all three configurations maintained the sensor on target for effectively the entire flight using the default HackHD lens. If a very narrow field of view had been used (25mm lens), there would have been a decrease in performance for the proposed FSM to 92%. Further analysis shows that if a smaller ground sample distance was desired, the final FSM design could have been flown with a field of view as low as 45° while still maintaining 100% time-on-target (assuming the same aspect ratio as the 25mm lens). The conclusion is that using the FSM did not sacrifice mission requirements to any significant degree in order to achieve increased flight path optimality. Figure 37 shows a screenshot taken from the gimbal mounted video collected during the real-world FSM flight test.

Figure 37: Screenshot from Real-World Ground Vehicle Tracking Mission

Final Firmware Modifications

Many modifications to the ArduPlane firmware were required for this research effort. The version used as a baseline was V2.68. From there, changes were made to the primary loop, the navigation process file, the telemetry management process file, and control mode response routines. Most noteworthy however, was the introduction of a new library and parameter modifications for control. Reference Appendix H for the final proposed ArduPlane firmware structure.

Ground Vehicle Class

To achieve the proposed heuristic behavior as well as conduct all described experimentation, it was necessary for the SUAS autopilot to access certain information regarding the ground vehicle being tracked. Because C++ and Arduino are object oriented languages, the most direct way of calculating, organizing, and presenting this data was to create a ground vehicle class, labeled Ground_Vehicle. Doing so allowed the main ArduPlane process to instantiate an object, notated GV, and when required call certain public attributes and methods. Using telemetry from the GCS, GV can be regularly updated to provide all pertinent information on the actual ground vehicle. This includes a safety check, GV.active, that allows the system to know if updates are no longer being received from the ground vehicle (even if a link with the GCS is still present) and failsafe to existing navigation logic. The Ground_Vehicle class was implemented using the traditional C++ library design [13], consisting of a header file called by ArduPlane, and an implementation file containing all logic associated with the defined methods. Figure 38 depicts a diagram showing the Ground_Vehicle class. Reference Appendix G for the associated C++ header.

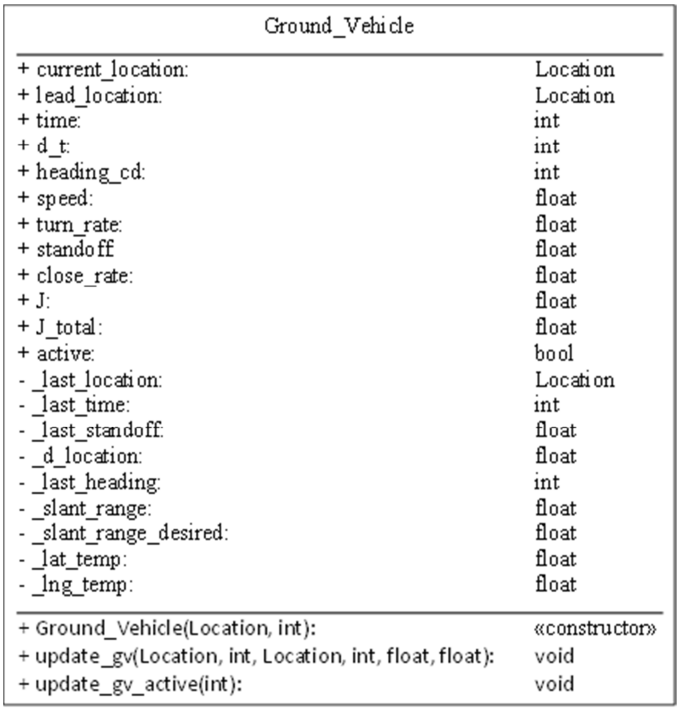

Figure 38: Class Diagram for Ground Vehicle

Parameter Entries

The number of flights required for this effort, even with most being conducted in HIL, was not feasible if every configuration change had required firmware adjustment, recompilation, uploading to the APM, power cycling, redoing the aircraft preflight, and reinitiating flight test. In order to conduct all requisite flights, especially during the experimentation stage of research, it was essential that the operator have real-time control over all factors. To address this challenge, changes made to the ArduPlane firmware, when possible, were parameterized and transmitted to the GCS upon connection. The final firmware version associated with this effort includes the following parameters in addition to the default configuration file. For future replication of any work, Appendices I and J define the parameter sets used for both real-world and HIL flights, respectively.

Target Tracking Mode

Tracking mode is a flag allowing the operator to enable or disable certain functionality. If desired, all code modifications associated with this effort could be turned off, resulting in reversion to entirely stock behavior. The second option is that only those changes listed in the initial modifications section (sensor gimbal tracking, and dual direction loiter) be enabled. The final option is to enable all altered functions, which was used during experimentation and FSM flights.

Lead Time

The lead time value is the number of seconds used when calculating a forward projected ground vehicle location. This was a key experimental parameter requiring real-time adjustment. The input value for lead time is passed into the GV object and handled internally, after which a public structure, labeled lead_location could be read and used for navigation.

Loiter Range

While using the loiter range for smooth transition to circular flight is a stock function, it was not made accessible to the operator by default. The range was a hard coded private attribute internal to the navigation process. Parameterization of this attribute allowed for the experimentation portion of the research to be executed as designed.

Loiter Direction

Allowing the loiter direction to be selected dynamically was an enabling function for the ground target tracking mission. However, this required that a defining parameter

be passed into the navigation process whenever a fixed loiter was required. This parameter allows direction specification for static loiter scenarios.

J_i Threshold

This parameter can be called by the navigation process and compared against the current J_i whenever assessing state-based behavior. Current J_i is a public attribute of GV used as a metric for state transition conditions in the proposed FSM design.

Control Effort Buffer

The control effort buffer, parameterized in meters, can be called when changing the level of effort applied to reach a desired slant range. This change was required for the alternate states proposed in the final FSM.

Summary

The analysis and results chapter expanded on the flight test methodology presented in Chapter 3 and describes in detail the results associated with each step. Initial discussion focused on firmware modifications made to enable the planned test procedures, including sensor gimbal target tracking and dynamic loiter direction. Next, results from the three planned test methodology phases were presented. Analysis was given as justification for performing a second iteration of the last two phases. These phases include settings experimentation and design of an FSM approach to heuristically approximate the proposed optimal path planning strategy. Flight data collected using the final recommended navigation logic was analyzed more extensively, providing evidence of statistically significant performance improvements. All test data was then presented

alongside the investigative questions by which each test was justified. Finally, an overview of the firmware modifications required to implement all proposed changes was discussed with focus on new object oriented structures and all entities implemented for user control. Chapter 5 will discuss the implications of these findings with attention to how results conclude the research objective, as well as recommendations for future work.

V. Conclusions and Recommendations

Chapter Overview

The final chapter concludes the research effort by expanding upon the data presented in Chapter 4 to discuss final implications as well as recommendations for follow-up action and future research. Conclusions focus on the stated research objective of approximating optimal flight path solutions for SUAS tracking of a mobile ground target. Follow-up actions are recommendations for work that could be done to augment the effort in order to validate or improve the achieved results. Finally, future research refers to potential work that could make use of the presented flight results or navigation strategy for other investigative purposes.

Conclusions of Research

The effort presented a research objective and four associated investigative questions. The first three questions formed the stages of research and focused on the characterization of achievable optimality for basic follow-me, DOE suggested, and state-based firmware configurations. Optimality was characterized using methods proposed by Livermore [1] for missions requiring a SUAS to track and monitor a moving ground target. The data collected during these stages is presented in Chapter 4 and it was concluded that each firmware setting, in the order conducted, achieved better average results than the previous.

The research objective was to achieve final implementation of the proposed strategy for approximation of optimal performance. The firmware was proposed and successfully implemented in the third stage of the effort. The final investigative question

was designed to characterize the implications of the applied firmware by describing the feasibility of achieving approximated optimality in real-world systems requiring autonomous mobile ground target tracking. To answer this question, the achieved cost of all executed flights was considered. The effort used data from a total of 47 flights at various settings: 16 for the first stage experiment, 10 for the second stage experiment, 4 using default follow-me, 3 at the initial DOE suggested settings, 3 with the initial FSM, 4 at the revised DOE suggested settings, 4 demonstrations of the final FSM, and 3 real-world flights. With regards to cost, the first quartile for all flights was found to be J = 4.915. The highest cost achieved by any of the four demonstration flights utilizing the final FSM was J = 3.178. In other words, flights with the final proposed firmware design fell within the best 25% of all results. Furthermore, while statistical analysis showed that the final FSM did not match the performance of the MATLAB generated optimal paths, it was found that cost was significantly improved over the baseline follow-me functionality. The relatively low cost of these flights coupled with the considerable performance increases over default capabilities indicate that near-optimal flight paths are operationally feasible using a real-time heuristic strategy implemented onboard the APM autopilot.

Follow-Up Action

Follow-up action describes potential efforts that could be done to improve on the documented results. These efforts would provide increased confidence in the presented findings and directly support the stated research objective.

Real-World Replication of Designed Experiment

One of the largest tradeoffs made in accomplishing this effort was the logistical inability to execute all flight tests using real-world equipment. All conducted simulation used a real APM physically connected to the GCS running both Mission Planner and the aircraft environment simulation software. The code being run was the actual APM firmware, as opposed to emulated software on the GCS. All navigation logic of concern in this effort was run without differentiation between real-world and simulated input states. This means that the experimental results are representative of real APM performance.

However, this does make it difficult to verify that the exact settings used are those that would work specifically on the real-world Rascal aircraft. For that reason, there would be some benefit to repeating the experimentation and demonstration portion of the effort in a real-world environment if possible. If constraints do not allow the entire 26 CCD flights to be executed, conducting smaller experiments to simply verify the factor limits and basic effects would also help to validate findings.

In addition, replication of experimentation should consider the possibility of using varied ground paths. The stated constraints for this effort allowed for only one path, which was selected to represent a range of tracking scenarios. However, this does not conclusively characterize universal performance. Validation of findings may be aided by examining a more exhaustive assortment of ground target paths.

Experimental Design to Analyze Finite State Machine

Designed experimentation was used in this effort to arrive at suggestions for existing (or easily modified) settings, which was consequently used for state analysis.

The final suggested FSM however, introduced two new factors (J_i threshold and control effort buffer) that were only flown at values concluded from initial analysis. If not resource constrained, it would be highly beneficial to perform a final DOE accounting for all pertinent parameters: loiter radius, loiter range, lead time, J_i threshold, and control effort buffer.

Lessons learned from the first attempt at analysis of experimental results showed that it can be difficult to detect and model both curvature and interrelationships of these parameters in a setting as complex as SUAS flight. Therefore, it would be suggested to begin with a screening experiment to determine which factors are truly significant and the approximate portion of the test space containing the best values for each. This could be conducted in a relatively efficient manner by beginning with a factorial (2^k) or fractional factorial (2^{k-p}) experiment using only a high and low setting for each factor. These types of experiments are commonly used for screening and have the potential to provide useful results when many factors are present and number of test points is limited [14]. After such a screener is conducted, more complex experimentation could be conducted in a narrower test space (with potentially fewer factors) to arrive at final suggested settings. It is possible that such an effort could better utilize the proposed navigation logic and achieve even better cost results.

Replication of Experimentation with Alternate Response

While the proposed FSM did result in significant performance increases, it was observed that the achieved path often had very little overlap with the associated optimal. Technically, the cost of a flight is the best measure of optimality, which is why J was selected as the primary response for all analysis in this effort. However, it is possible that

any further improvement over the proposed FSM may require consideration of horizontal deviation from the true optimal path.

The challenge of characterizing two dimensional path deviation between any two sets of aircraft flight data was discussed by McCarthy in his attempt to analyze close-formation flight capability for SUAS [15]. Specifically, he was attempting to achieve formation flight using a dynamic waypoint update strategy. When addressing the feasibility of such an approach, McCarthy recognized that adherence to waypoint paths requires comparative characterization to identify the best achieved autopilot parameter set. His solution was a MATLAB script capable of comparing two location matrices and charting XY deviation against a normalized time vector. This deviation, while not a direct measure of the optimality with which the current effort is concerned, still provides useful comparative information regarding the similarity of any two flight paths. Figure 39 shows an example two dimensional path comparison generated by McCarthy. Figure 40 is the associated chart showing horizontal path deviation. Parameterization of this deviation and use as an alternate or secondary response in experimentation may allow even more significant cost improvements with a heuristic real-time strategy.

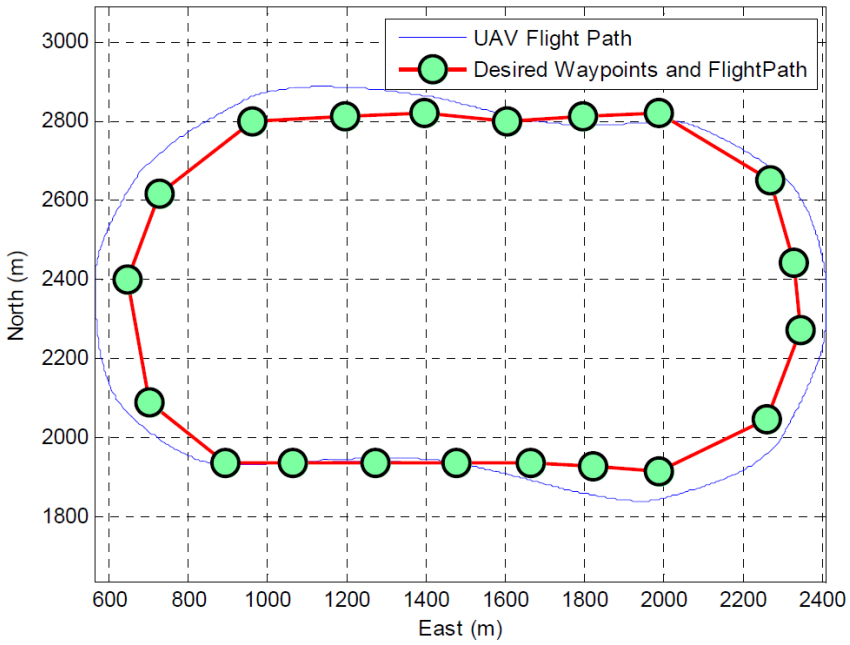

Figure 39: McCarthy Example Flight Path Visualization

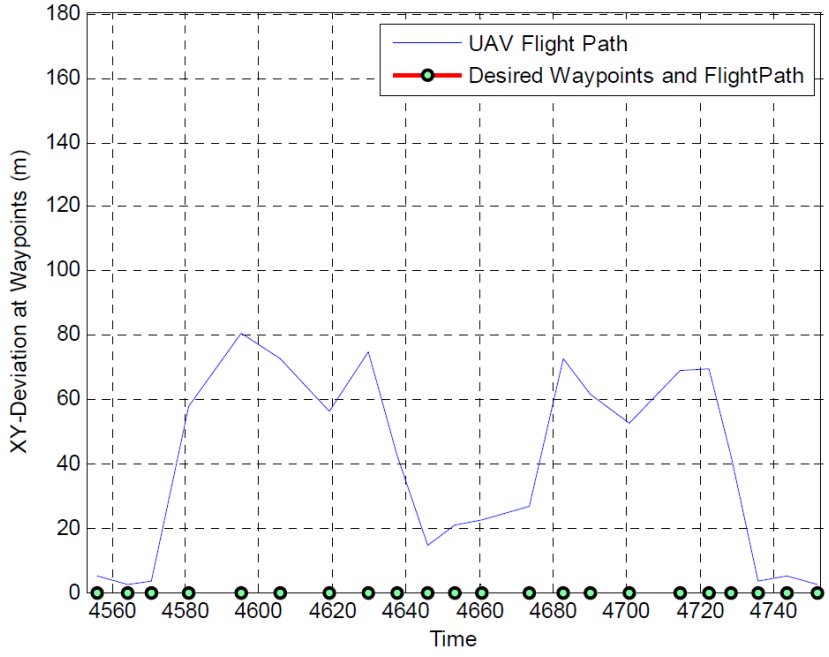

Figure 40: McCarthy Example Flight Path Deviation Chart

Future Research

Future research suggests alternate efforts that may benefit from the documented results. These suggestions are for work that does not directly support the stated research objective, but rather focuses on new objectives in related areas.

Analysis of Optimization Cost Function

This effort attempted to achieve the lowest possible objective cost function value, as defined by Livermore [1], in a heuristic, real-time fashion. However, no investigation was given to the value of the cost function as a measurement of convoy overwatch performance. Future work in the field of flight path optimization, specifically flights aimed at mobile target tracking, would benefit from validation of the cost metric through the application of systems engineering principles.

To achieve this validation, a true requirements elicitation should be conducted for the convoy overwatch mission including, but not limited to, input from those conducting the ground missions as well as those performing intelligence processing. The results of such an effort would include, as a subset, any technical requirements associated with conducting convoy overwatch with a SUAS. Any given flight path alternatives, theoretical or real-world, can be compared against the key performance parameters associated with such requirements and rank ordered using traditional decision analysis. Ranking in such a manner can help validate the cost function used for this effort as the achieved J values, when sorted, should align with the decision analysis results. Furthermore, any future proposed cost function can be validated in the same manner.

Stochastic Estimation of Ground Vehicle Path

The final recommendation for future research focuses on prediction of the path of the ground vehicle. In the case of this research effort, it is assumed that operations are occurring in an environment where future knowledge of the ground path is not feasible. The implementation of the lead time functionality accounts for only the current ground vehicle heading and speed to predict a linear future location.

In Livermore's work, he finds that it is unnecessary to know the entire future path of the ground vehicle to arrive at a feasible optimal path. He presents a strategy by which the optimization function is called repeatedly (at 1.5Hz) considering the current states of the air and ground vehicle as well as the future path for only a specified period of time (labeled as the look-ahead). The notional air vehicle executes the first 0.667s of the returned flight path before reevaluating. He finds that using a look-ahead as low as 4 seconds for the future ground path knowledge results in an overall flight effectively identical in path and cost to a single iteration of the optimization function considering the full future path [1].

If future efforts or constraint changes allow for updates to the Ground_Vehicle library that provide an estimation of the future path for as little as 4 seconds, it is possible that real-time execution of Livermore's path planning strategy could be implemented in a non-heuristic fashion. If the period of time is small enough, work could be done to implement a nonlinear optimization C++ library into the ArduPlane firmware to most thoroughly emulate Livermore's strategy. Conversely, if true onboard optimization libraries prove computationally excessive, simple functions can be written to consider a fixed number of look-ahead flight paths and return the lowest cost option as either a

waypoint array or direct control sequence. Both options have the potential to realize significant performance benefits in terms of onboard approaches to optimal path planning.

Summary

The presented research evaluated the feasibility of achieving heuristic path planning strategies running in real-time onboard a SUAS performing a convoy overwatch mission. The proposed strategy was designed to emulate, to the best extent possible, an existing flight path optimization function built for post-processing assuming full future ground vehicle information. Work began by evaluating the default behavior of the APM autopilot. Minor modifications were made to parameterize existing settings as well as add basic functionality that previous research suggested to be important. Changes included adjustments to the sensor gimbal control library, addition of a dynamic loiter direction, and the ability to lead the ground vehicle by a given time period.

Next, a two stage designed experiment was conducted to arrive at the best achievable combination of settings (with regards to flight path optimality). A time analysis of instantaneous contributions to optimality (J_i) was performed and a finite state machine approach to navigation logic was proposed to further increase performance. The suggested FSM was integrated into the APM flight firmware and tested in a six-degree-of-freedom hardware in the loop environment. It was found that achieved optimality demonstrated a statistically significant improvement over the default follow-me performance. The effort concludes that real-time heuristic approximations to optimal

path planning do present a viable alternative to the high computational and equipment costs associated with implementing a true optimal solution.

REFERENCES

[1] R. Livermore, "Optimal UAV Path Planning for Tracking a Moving Ground Vehicle with a Gimbaled Camera," Air Force Institute of Technology, Wright Patterson Air Force Base, 2014.

[2] B. D. Lozano, "Improving Unmanned Aircraft Persistance by Enhancing Endurance and Effective Surveillance Using Design of Experiments and Regression Analysis," Air Force Institute of Technology, Wright-Patterson Air Force Base, 2011.

[3] N. A. Terning, "Real-Time Navigation and Flight Path Generation for Tracking Stop-and-Go Targets with Miniature Air Vehicles," Air Force Institute of Technology, Wright-Patterson Air Force Base, 2008.

[4] J. P. Boire, "Autonomous Routing of Unmanned Aerial Vehicle (UAV) Relays to Mimic Optimal Trajectories in Real Time," Air Force Institue of Technology, Wright-Patterson Air Force Base, 2011.

[5] Headquarters, United States Air Force, "United States Air Force Unmanned Aircraft Systems Flight Plan 2009-2047," Washington DC, 2009.

[6] RAND Corporation, Arroyo Center, "Unmanned Aircraft Systems for Logistics Applications," RAND Corporation, Santa Monica, 2011.

[7] M. D. Zollars, "Optimal Wind Corrected Flight Path Planning for Autonomous Micro Air Vehicles," Air Force Institute of Technology, Wright-Patterson Air Force Base, 2007.

[8] J. W. Welborn, "Calibration and Extension of a Discrete Event Operations Simulation Modeling Multiple Un-Manned Aerial Vehicles Controlled by a Single Operator," Air Force Institute of Technology, Wright-Patterson Air Force Base, 2013.

[9] J. Gundlach, Designing Unmanned Aircraft Systems: A Comprehensive Approach, Reston: American Institute of Aeronautics and Astronautics, 2012.

[10] "ArduPilot instructional graphics source," DIYDrones, 28 November 2012. [Online]. Available: https://code.google.com/p/ardupilot-mega/wiki/APM2_Graphics. [Accessed 1 August 2013].

[11] J. Berndt, "JSBSim: An Open Source, Platform-Independent, Flight Dynamics Model in C++," 2011. [Online]. Available: jsbsim.sourceforge.net/JSBSimReferenceManual.pdf. [Accessed 17 February 2014].

[12] "diydrones/ardupilot · GitHub," [Online]. Available: https://github.com/diydrones/ardupilot. [Accessed 9 March 2014].

[13] S. Monk, Programming Arduino Next Steps: Going Further with Sketches, New York: McGraw-Hill Education, 2014.

[14] D. Montgomery, Design and Analysis of Experiments, Hoboken, NJ: John Wiley & Sons, 2013.

[15] P. A. McCarthy, "Characterization of UAV Performance and Development of a Formation Flight Controller for Multiple Small UAVs," Air Force Institute of Technology, Wright-Patterson Air Force Base, 2006.

[16] "3DRobotics - Learn," 3DRobotics, 2013. [Online]. Available: http://3drobotics.com/learn/#APM_26_Autopilot. [Accessed 11 February 2014].

[17] "SPT100 Pan & Tilt System," Servocity, 2014. [Online]. Available: http://www.servocity.com/html/spt100_pan___tilt_system.html. [Accessed 11 February 2014].

[18] "HackHD - Technical Specifications," HackHD, 2013. [Online]. Available: http://www.hackhd.com/tech.php. [Accessed 11 February 2014].

Appendix A: Rascal Configuration

Figure 41: Rascal SUAS Used for Flight Test

Table 18: Rascal SUAS Key Specifications

Wingspan	110 in.
Wing Area	1522 sq. in.
Length	75.75 in.
Flying Weight	≈ 12 lbs.
Propeller	APC 18x8E
Motor	Himax HC6330-200
Electronic Speed Control	Castle 120A HV
Flight Batteries	Turnigy 5000 mAh LiPo
Autopilot	ArduPilot Mega 2.5
Cruise Airspeed	15 m/s
Aileron Servos	Hitec HS-6635HB
Aileron Deflection	±27°
Elevator Servo	Hitec HS-5485HB
Elevator Deflection	±19°
Rudder Servo	Hitec HS-5485HB
Rudder Deflection	±12°
Maximum Roll Rate	100°/sec.

Appendix B: Autopilot and Peripherals Specifications

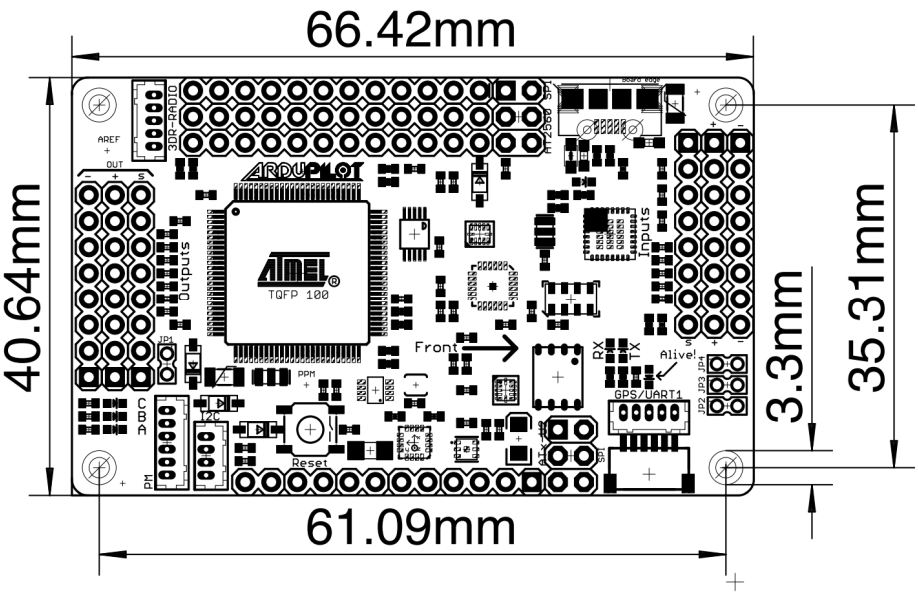

Figure 42: APM 2.5 Dimensions [16]

Table 19: Autopilot Specifications

Autopilot	ArduPilot Mega
Hardware Version	2.5
Software Version	2.68 with modifications
Processor	Atmel 2560
Gyro + Accelerometer	InvenSense MPU-6000
Magnetometer	Honeywell HMC5883L
Barometric Sensor	Measurement Specialties MS5611-01BA03
GPS Receiver	uBlox LEA-6H
Airspeed Sensor	Freescale MPXV7002
Telemetry Modem	3DRobotics Radio Set

Table 20: Telemetry Modem Specifications

Modem Brand	3DRobotics
Frequency	915 MHz
Transmission Type	Frequency Hopping Spread Spectrum
Data Connection	6 Pin DF13
Maximum Output Power	100 mW
Rx Sensitivity	-117 dBm
Transmission Connector	RP-SMA
Supply Voltage	3.7-6 VDC
Size	26.7 x 55.5 x 13.3mm

Appendix C: Ground Control Station Specifications

Table 21: Ground Control Station Equipment

Computer	HP EliteBook 8560w
Ground Control Software	APM Mission Planner
Software Version	1.2.76
Telemetry Modem	3DRobotics Radio Set
GPS Receiver	GlobalSat BU-353
Ground Vehicle	HMMWV Troop Carrier Configuration

Appendix D: Payload Specifications

Table 22: Payload Components

Gimbal	
Model	Servocity SPT100H
Pan Servo	Hitec HS-785HB
Pan Rotation	±180°
Pan Pulsewidth Range	1390-1625
Tilt Servo	Hitec HS-5485HB
Tilt Rotation	+10°, -90°
Tilt Pulsewidth Range	1000-2000
Camera	
Model	HackHD
Resolution	1080P
Pixel Count	9MP
Framerate	30 FPS
Aspect Ratio	16:9
Storage	onboard microSD
Lens Mount	M12
Video Output	Composite 480P
Supply Voltage	3.7-5 VDC
Transmission	
Frequency	5.8 GHz
Transmitter	ImmersionRC TX_5G8_600
Tx Power	600 mW
Supply Voltage	6-25 VDC
Receiver	Iftron Yellowjacket Diversity
Supply Voltage	6-15 VDC
Rx Sensitivity	-91 dBm

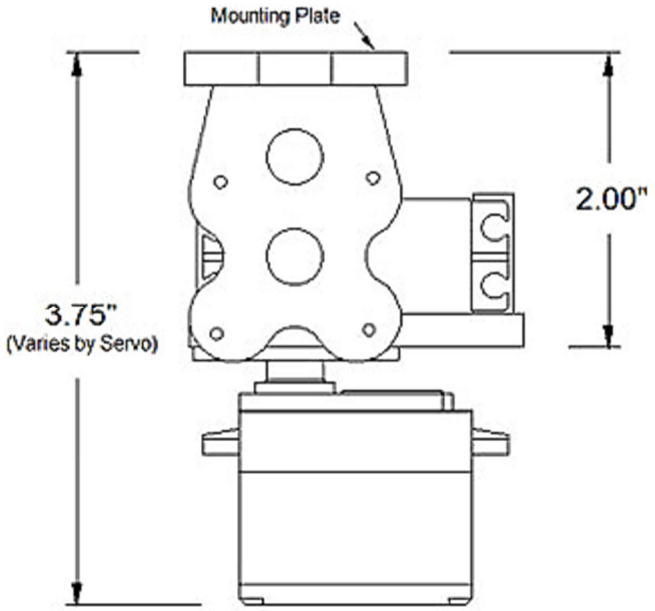

Figure 43: Servocity SPT100H Pan-Tilt Gimbal Dimensional Drawing [17]

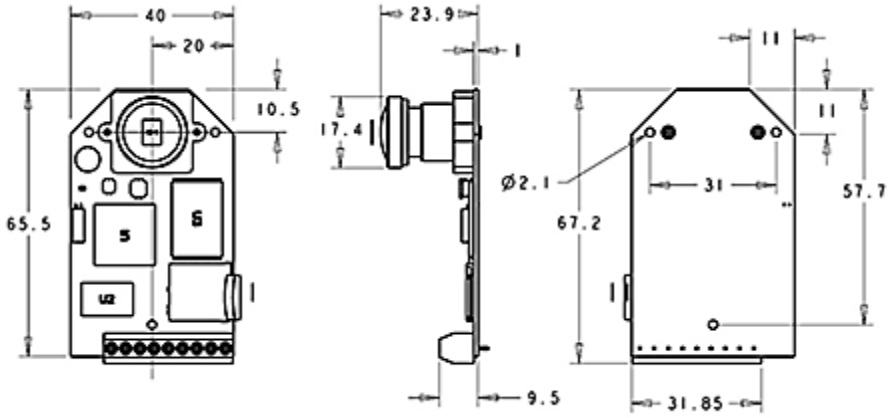

Figure 44: HackHD Camera Dimensional Drawing [18]

Appendix E: Simulated Rascal Definition

```
<?xml version="1.0"?>
<?xml-stylesheet
href="http://jsbsim.sourceforge.net/JSBSim.xsl"
type="text/xsl"?>
<fdm_config name="rascal" version="2.0" release="BETA"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="http://jsbsim.sourceforge.n
et/JSBSim.xsd">

    <fileheader>
        <author> Author Name </author>
        <filecreationdate> Creation Date
</filecreationdate>
        <version> Version </version>
        <description> Models a rascal </description>
    </fileheader>

    <metrics>
        <wingarea unit="FT2"> 10.57 </wingarea>
        <wingspan unit="FT"> 9.17 </wingspan>
        <chord unit="FT"> 1.15 </chord>
        <htailarea unit="FT2"> 1.69 </htailarea>
        <htailarm unit="FT"> 3.28 </htailarm>
        <vtailarea unit="FT2"> 1.06 </vtailarea>
        <vtailarm unit="FT"> 0 </vtailarm>
        <location name="AERORP" unit="IN">
            <x> 37.4 </x>
            <y> 0 </y>
            <z> 0 </z>
        </location>
        <location name="EYEPOINT" unit="IN">
            <x> 20 </x>
            <y> 0 </y>
            <z> 5 </z>
        </location>
        <location name="VRP" unit="IN">
            <x> 0 </x>
            <y> 0 </y>
            <z> 0 </z>
        </location>
    </metrics>

    <mass_balance>
        <ixx unit="SLUG*FT2"> 1.95 </ixx>
        <iyy unit="SLUG*FT2"> 1.55 </iyy>
        <izz unit="SLUG*FT2"> 1.91 </izz>
        <ixy unit="SLUG*FT2"> 0 </ixy>
        <ixz unit="SLUG*FT2"> 0 </ixz>
        <iyz unit="SLUG*FT2"> 0 </iyz>
        <emptywt unit="LBS"> 13 </emptywt>
        <location name="CG" unit="IN">
            <x> 36.4 </x>
            <y> 0 </y>
            <z> 4 </z>
        </location>
    </mass_balance>

    <ground_reactions>
        <contact type="BOGEY" name="LEFT_MLG">
            <location unit="IN">
                <x> 33.1 </x>
                <y> -12.9 </y>
                <z> -13.1 </z>
            </location>
            <static_friction> 0.8 </static_friction>
            <dynamic_friction> 0.5 </dynamic_friction>
            <rolling_friction> 0.1 </rolling_friction>
            <spring_coeff unit="LBS/FT"> 480
</spring coeff>
            <damping_coeff unit="LBS/FT/SEC"> 100
</damping_coeff>
            <max_steer unit="DEG"> 0.0 </max_steer>
            <brake_group> NONE </brake_group>
            <retractable>0</retractable>
        </contact>
        <contact type="BOGEY" name="RIGHT_MLG">
            <location unit="IN">
                <x> 33.1 </x>
                <y> 12.9 </y>
                <z> -13.1 </z>
            </location>
            <static_friction> 0.8 </static_friction>
            <dynamic_friction> 0.5 </dynamic_friction>
            <rolling_friction> 0.1 </rolling_friction>
            <spring_coeff unit="LBS/FT"> 480
</spring_coeff>
            <damping_coeff unit="LBS/FT/SEC"> 100
</damping_coeff>
            <max_steer unit="DEG"> 0.0 </max_steer>
            <brake_group> NONE </brake_group>
            <retractable>0</retractable>
        </contact>
        <contact type="BOGEY" name="TAIL_LG">
            <location unit="IN">
                <x> 68.9 </x>
                <y> 0 </y>
                <z> -13.1 </z>
            </location>
            <static_friction> 8.0 </static_friction>
            <dynamic_friction> 5.0 </dynamic_friction>
            <rolling_friction> 0.1 </rolling_friction>
            <spring_coeff unit="LBS/FT"> 480
</spring coeff>
            <damping_coeff unit="LBS/FT/SEC"> 100
</damping_coeff>
            <max_steer unit="DEG"> 360.0 </max_steer>
            <brake_group> NONE </brake_group>
            <retractable>0</retractable>
        </contact>
    </ground_reactions>

    <propulsion>
        <engine file="Zenoah_G-26A">
            <location unit="IN">
                <x> 36 </x>
                <y> 0 </y>
                <z> 0 </z>
            </location>
            <orient unit="DEG">
                <roll> 0.0 </roll>
                <pitch> 0 </pitch>
                <yaw> 0 </yaw>
            </orient>
            <feed>0</feed>
            <thruster file="18x8">
                <location unit="IN">
                    <x> 1 </x>
                    <y> 0 </y>
                    <z> 0 </z>
                </location>
                <orient unit="DEG">
                    <roll> 0.0 </roll>
                    <pitch> 0.0 </pitch>
                    <yaw> 0.0 </yaw>
                </orient>
                <p_factor>1.0</p_factor>
            </thruster>
        </engine>
        <tank type="FUEL">    <!-- Tank number 0 -->
            <location unit="IN">
                <x> 36.36 </x>
                <y> 0 </y>
                <z> -1.89375 </z>
            </location>
            <capacity unit="LBS"> 1.5 </capacity>
            <contents unit="LBS"> 1.5 </contents>
        </tank>
    </propulsion>
```

```
<flight_control name="FCS: rascal">
 <channel name="All">

    <summer name="Pitch Trim Sum">
        <input>fcs/elevator-cmd-norm</input>
        <input>fcs/pitch-trim-cmd-norm</input>
        <clipto>
            <min>-1</min>
            <max>1</max>
        </clipto>
    </summer>

    <aerosurface_scale name="Elevator Control">
        <input>fcs/pitch-trim-sum</input>
        <range>
            <min>-0.35</min>
            <max>0.3</max>
        </range>
        <output>fcs/elevator-pos-rad</output>
    </aerosurface_scale>

    <aerosurface_scale name="Elevator Normalized">
        <input>fcs/elevator-pos-rad</input>
        <domain>
            <min>-0.3</min>
            <max> 0.3</max>
        </domain>
        <range>
            <min>-1</min>
            <max> 1</max>
        </range>
        <output>fcs/elevator-pos-norm</output>
    </aerosurface_scale>

    <summer name="Roll Trim Sum">
        <input>fcs/aileron-cmd-norm</input>
        <input>fcs/roll-trim-cmd-norm</input>
        <clipto>
            <min>-1</min>
            <max>1</max>
        </clipto>
    </summer>

    <aerosurface_scale name="Left Aileron Control">
        <input>fcs/roll-trim-sum</input>
        <range>
            <min>-0.35</min>
            <max>0.35</max>
        </range>
        <output>fcs/left-aileron-pos-rad</output>
    </aerosurface_scale>

    <aerosurface_scale name="Right Aileron Control">
        <input>-fcs/roll-trim-sum</input>
        <range>
            <min>-0.35</min>
            <max>0.35</max>
        </range>
        <output>fcs/right-aileron-pos-rad</output>
    </aerosurface_scale>

    <aerosurface_scale name="Left aileron Normalized">
        <input>fcs/left-aileron-pos-rad</input>
        <domain>
            <min>-0.35</min>
            <max> 0.35</max>
        </domain>
        <range>
            <min>-1</min>
            <max> 1</max>
        </range>
        <output>fcs/left-aileron-pos-norm</output>
    </aerosurface_scale>

    <aerosurface_scale name="Right aileron
Normalized">
        <input>fcs/right-aileron-pos-rad</input>
        <domain>
            <min>-0.35</min>
            <max> 0.35</max>
        </domain>
        <range>
            <min>-1</min>
            <max> 1</max>
        </range>
        <output>fcs/right-aileron-pos-norm</output>
    </aerosurface_scale>

    <summer name="Rudder Command Sum">
        <input>fcs/rudder-cmd-norm</input>
        <input>fcs/yaw-trim-cmd-norm</input>
        <clipto>
            <min>-1</min>
            <max>1</max>
        </clipto>
    </summer>

    <aerosurface_scale name="Rudder Control">
        <input>fcs/rudder-command-sum</input>
        <range>
            <min>-0.35</min>
            <max>0.35</max>
        </range>
        <output>fcs/rudder-pos-rad</output>
    </aerosurface_scale>

    <aerosurface_scale name="Rudder Normalized">
        <input>fcs/rudder-pos-rad</input>
        <domain>
            <min>-0.35</min>
            <max> 0.35</max>
        </domain>
        <range>
            <min>-1</min>
            <max> 1</max>
        </range>
        <output>fcs/rudder-pos-norm</output>
    </aerosurface_scale>
 </channel>
</flight_control>

<aerodynamics>
    <axis name="DRAG">
        <function name="aero/coefficient/CD0">

<description>Drag_at_zero_lift</description>
            <product>
                <property>aero/qbar-psf</property>
                <property>metrics/Sw-sqft</property>
                  <table>
                      <independentVar>aero/alpha-
rad</independentVar>
                      <tableData>
                          -1.5700 1.5000
                          -0.2600 0.0560
                          0.0000  0.0280
                          0.2600  0.0560
                          1.5700  1.5000
                      </tableData>
                  </table>
            </product>
        </function>
        <function name="aero/coefficient/CDi">
            <description>Induced_drag</description>
            <product>
                <property>aero/qbar-psf</property>
                <property>metrics/Sw-sqft</property>
                <property>aero/cl-squared</property>
                <value>0.0400</value>
            </product>
        </function>
        <function name="aero/coefficient/CDbeta">

<description>Drag_due_to_sideslip</description>
            <product>
                <property>aero/qbar-psf</property>
                <property>metrics/Sw-sqft</property>
                  <table>
```

```
                        <independentVar>aero/beta-
rad</independentVar>
                        <tableData>
                            -1.5700 1.2300
                            -0.2600 0.0500
                            0.0000  0.0000
                            0.2600  0.0500
                            1.5700  1.2300
                        </tableData>
                    </table>
                </product>
            </function>
            <function name="aero/coefficient/CDde">

<description>Drag_due_to_Elevator_Deflection</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>fcs/elevator-pos-
norm</property>
                    <value>0.0300</value>
                </product>
            </function>
        </axis>

        <axis name="SIDE">
            <function name="aero/coefficient/CYb">

<description>Side_force_due_to_beta</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>aero/beta-rad</property>
                    <value>-1.0000</value>
                </product>
            </function>
        </axis>

        <axis name="LIFT">
            <function name="aero/coefficient/CLalpha">

<description>Lift_due_to_alpha</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                      <table>
                        <independentVar>aero/alpha-
rad</independentVar>
                        <tableData>
                            -0.2000 -0.7500
                            0.0000  0.2500
                            0.2300  1.4000
                            0.6000  0.7100
                        </tableData>
                    </table>
                </product>
            </function>
            <function name="aero/coefficient/CLde">

<description>Lift_due_to_Elevator_Deflection</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>fcs/elevator-pos-
rad</property>
                    <value>0.2000</value>
                </product>
            </function>
        </axis>

        <axis name="ROLL">
            <function name="aero/coefficient/Clb">

<description>Roll_moment_due_to_beta</description>
                <!-- aka dihedral effect -->
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/bw-ft</property>
                    <property>aero/beta-rad</property>
                    <value>-0.1000</value>
                </product>
            </function>
            <function name="aero/coefficient/Clp">

<description>Roll_moment_due_to_roll_rate</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/bw-ft</property>
                    <property>aero/bi2vel</property>
                    <property>velocities/p-aero-
rad_sec</property>
                    <value>-0.4000</value>
                </product>
            </function>
            <function name="aero/coefficient/Clr">

<description>Roll_moment_due_to_yaw_rate</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/bw-ft</property>
                    <property>aero/bi2vel</property>
                    <property>velocities/r-aero-
rad_sec</property>
                    <value>0.1500</value>
                </product>
            </function>
            <function name="aero/coefficient/Clda">

<description>Roll_moment_due_to_aileron</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/bw-ft</property>
                    <property>fcs/left-aileron-pos-
rad</property>
                      <table>

<independentVar>velocities/mach</independentVar>
                        <tableData>
                            0.0000  0.1300
                            2.0000  0.0570
                        </tableData>
                    </table>
                </product>
            </function>
            <function name="aero/coefficient/Cldr">

<description>Roll_moment_due_to_rudder</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/bw-ft</property>
                    <property>fcs/rudder-pos-
rad</property>
                    <value>0.0100</value>
                </product>
            </function>
        </axis>

        <axis name="PITCH">
            <function name="aero/coefficient/Cmalpha">

<description>Pitch_moment_due_to_alpha</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/cbarw-ft</property>
                    <property>aero/alpha-rad</property>
                    <value>-0.5000</value>
                </product>
            </function>
            <function name="aero/coefficient/Cmde">

<description>Pitch_moment_due_to_elevator</description>
                <product>
```

```
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/cbarw-ft</property>
                    <property>fcs/elevator-pos-
rad</property>
                      <table>

<independentVar>velocities/mach</independentVar>
                        <tableData>
                            0.0000  -0.5000  <!-- was -
1.1 -->
                            2.0000  -0.2750
                        </tableData>
                      </table>
                </product>
            </function>
            <function name="aero/coefficient/Cmq">

<description>Pitch_moment_due_to_pitch_rate</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/cbarw-ft</property>
                    <property>aero/ci2vel</property>
                    <property>velocities/q-aero-
rad_sec</property>
                    <value>-12.0000</value>
                </product>
            </function>
            <function name="aero/coefficient/Cmadot">

<description>Pitch_moment_due_to_alpha_rate</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/cbarw-ft</property>
                    <property>aero/ci2vel</property>
                    <property>aero/alphadot-
rad_sec</property>
                    <value>-7.0000</value>
                </product>
            </function>
        </axis>

        <axis name="YAW">
            <function name="aero/coefficient/Cnb">

<description>Yaw_moment_due_to_beta</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/bw-ft</property>
                    <property>aero/beta-rad</property>
                    <value>0.1200</value>
                </product>
            </function>
            <function name="aero/coefficient/Cnr">

<description>Yaw_moment_due_to_yaw_rate</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/bw-ft</property>
                    <property>aero/bi2vel</property>
                    <property>velocities/r-aero-
rad_sec</property>
                    <value>-0.1500</value>
                </product>
            </function>
            <function name="aero/coefficient/Cndr">

<description>Yaw_moment_due_to_rudder</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/bw-ft</property>
                    <property>fcs/rudder-pos-
rad</property>
                    <value>-0.0500</value>
                </product>
            </function>
            <function name="aero/coefficient/Cnda">
                <description>Adverse_yaw</description>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/bw-ft</property>
                    <property>fcs/left-aileron-pos-
rad</property>
                    <value>-0.0300</value>
                </product>
            </function>
            <function name="aero/coefficient/Cndi">

<description>Yaw_moment_due_to_tail_incidence</description
>
                <product>
                    <property>aero/qbar-psf</property>
                    <property>metrics/Sw-sqft</property>
                    <property>metrics/bw-ft</property>
                    <value>0.0007</value>
                </product>
            </function>
        </axis>
    </aerodynamics>
</fdm_config>
```

Appendix F: AP_Mount Revised update_mount_position Function

```
/// This one should be called periodically
void AP_Mount::update_mount_position(struct Location *guided_WP_target, bool
guided_mode_bool)
//added input arguments Jul13, ref. AP_Mount.h  -cjn
{
#if MNT_RETRACT_OPTION == ENABLED
    static bool mount_open = 0;     // 0 is closed
#endif

    switch((enum MAV_MOUNT_MODE)_mount_mode.get())
    {
#if MNT_RETRACT_OPTION == ENABLED
    // move mount to a "retracted position" or to a position where a fourth servo can
retract the entire mount into the fuselage
    case MAV_MOUNT_MODE_RETRACT:
    {
        Vector3f vec = _retract_angles.get();
        _roll_angle  = vec.x;
        _tilt_angle  = vec.y;
        _pan_angle   = vec.z;
        break;
    }
#endif

    // move mount to a neutral position, typically pointing forward
    case MAV_MOUNT_MODE_NEUTRAL:
    {
        Vector3f vec = _neutral_angles.get();
        _roll_angle  = vec.x;
        _tilt_angle  = vec.y;
        _pan_angle   = vec.z;
        break;
    }

    // point to the angles given by a mavlink message
    case MAV_MOUNT_MODE_MAVLINK_TARGETING:
    {
        Vector3f vec = _control_angles.get();
        _roll_control_angle  = radians(vec.x);
        _tilt_control_angle  = radians(vec.y);
        _pan_control_angle   = radians(vec.z);
        stabilize();
        break;
    }

    // RC radio manual angle control, but with stabilization from the AHRS
    case MAV_MOUNT_MODE_RC_TARGETING:
    {
#if MNT_JSTICK_SPD_OPTION == ENABLED
        if (_joystick_speed) {                  // for spring loaded joysticks
            // allow pilot speed position input to come directly from an RC_Channel
            if (_roll_rc_in && (rc_ch[_roll_rc_in-1])) {
                _roll_control_angle += rc_ch[_roll_rc_in-1]->norm_input() * 0.00001 *
                _joystick_speed;
                if (_roll_control_angle < radians(_roll_angle_min*0.01))
                _roll_control_angle = radians(_roll_angle_min*0.01);
                if (_roll_control_angle > radians(_roll_angle_max*0.01))
                _roll_control_angle = radians(_roll_angle_max*0.01);
            }
            if (_tilt_rc_in && (rc_ch[_tilt_rc_in-1])) {
                _tilt_control_angle += rc_ch[_tilt_rc_in-1]->norm_input() * 0.00001 *
                _joystick_speed;
                if (_tilt_control_angle < radians(_tilt_angle_min*0.01))
                _tilt_control_angle = radians(_tilt_angle_min*0.01);
                if (_tilt_control_angle > radians(_tilt_angle_max*0.01))
                _tilt_control_angle = radians(_tilt_angle_max*0.01);
```

```
            }
            if (_pan_rc_in && (rc_ch[_pan_rc_in-1])) {
                _pan_control_angle += rc_ch[_pan_rc_in-1]->norm_input() * 0.00001 *
                _joystick_speed;
                if (_pan_control_angle < radians(_pan_angle_min*0.01)) _pan_control_angle
                = radians(_pan_angle_min*0.01);
                if (_pan_control_angle > radians(_pan_angle_max*0.01)) _pan_control_angle
                = radians(_pan_angle_max*0.01);
            }
        } else {
#endif
            // allow pilot position input to come directly from an RC_Channel
            if (_roll_rc_in && (rc_ch[_roll_rc_in-1])) {
                _roll_control_angle = angle_input_rad(rc_ch[_roll_rc_in-1],
                _roll_angle_min, _roll_angle_max);
            }
            if (_tilt_rc_in && (rc_ch[_tilt_rc_in-1])) {
                _tilt_control_angle = angle_input_rad(rc_ch[_tilt_rc_in-1],
                _tilt_angle_min, _tilt_angle_max);
            }
            if (_pan_rc_in && (rc_ch[_pan_rc_in-1])) {
                _pan_control_angle = angle_input_rad(rc_ch[_pan_rc_in-1], _pan_angle_min,
                _pan_angle_max);
            }
#if MNT_JSTICK_SPD_OPTION == ENABLED
        }
#endif
        stabilize();
        break;
    }

#if MNT_GPSPOINT_OPTION == ENABLED
    // point mount to a GPS point given by the mission planner
    case MAV_MOUNT_MODE_GPS_POINT:
    {
        if(_gps->fix) {
                        //if in guided mode, calls calc_GPS_target_angle with guided
                        waypoint location
                        if (guided_mode_bool==1) {
                                calc_GPS_target_angle(guided_WP_target);
                                }
                                else {
                                calc_GPS_target_angle(&_target_GPS_location);
                                }
            stabilize();
        }
        break;
    }
#endif

    default:
        break;
    }
```

Appendix G: Ground_Vehicle Library Definition

```
/*****************************************************
Ground_Vehicle.h:   library for ground vehicle class
Author:             Neal, Charles
Date:               January 2014
Purpose:            keeps track of ground vehicle being
                    updated using follow-me mode.
*****************************************************/

#ifndef Ground_Vehicle_h
#define Ground_Vehicle_h

#include "Arduino.h"
#include <AP_Common.h>
#include <AP_Math.h>

class Ground_Vehicle
{
  public:

    //Constructor
    Ground_Vehicle(Location start_location, int start_time);

    //Update all GV attributes
    void update_gv(Location new_location, int new_time, Location AC_location, int
desired_radius, float alt, float lead_time);

    //Call frequently to update the active flag
    void update_gv_active(int check_time);

    // Public Attributes
    Location        current_location;
    Location        lead_location;
    int             time;   //milliseconds from millis()
    int             d_t;    //milliseconds
    int             heading_cd; //centi-degrees
    float           speed;  //m/s
    float           turn_rate;  //deg/s
    float           standoff;   //meters
    float           close_rate; //m/s
    float           J;
    float           J_total;
    bool            active;

  private:

    // Private Attributes
    struct Location _last_location;
    int             _last_time;
    float           _last_standoff;
    float           _d_location;
    int             _last_heading;
    float           _slant_range;
    float           _slant_range_desired;
    float           _lat_temp;
    float           _lng_temp;

};

#endif
```

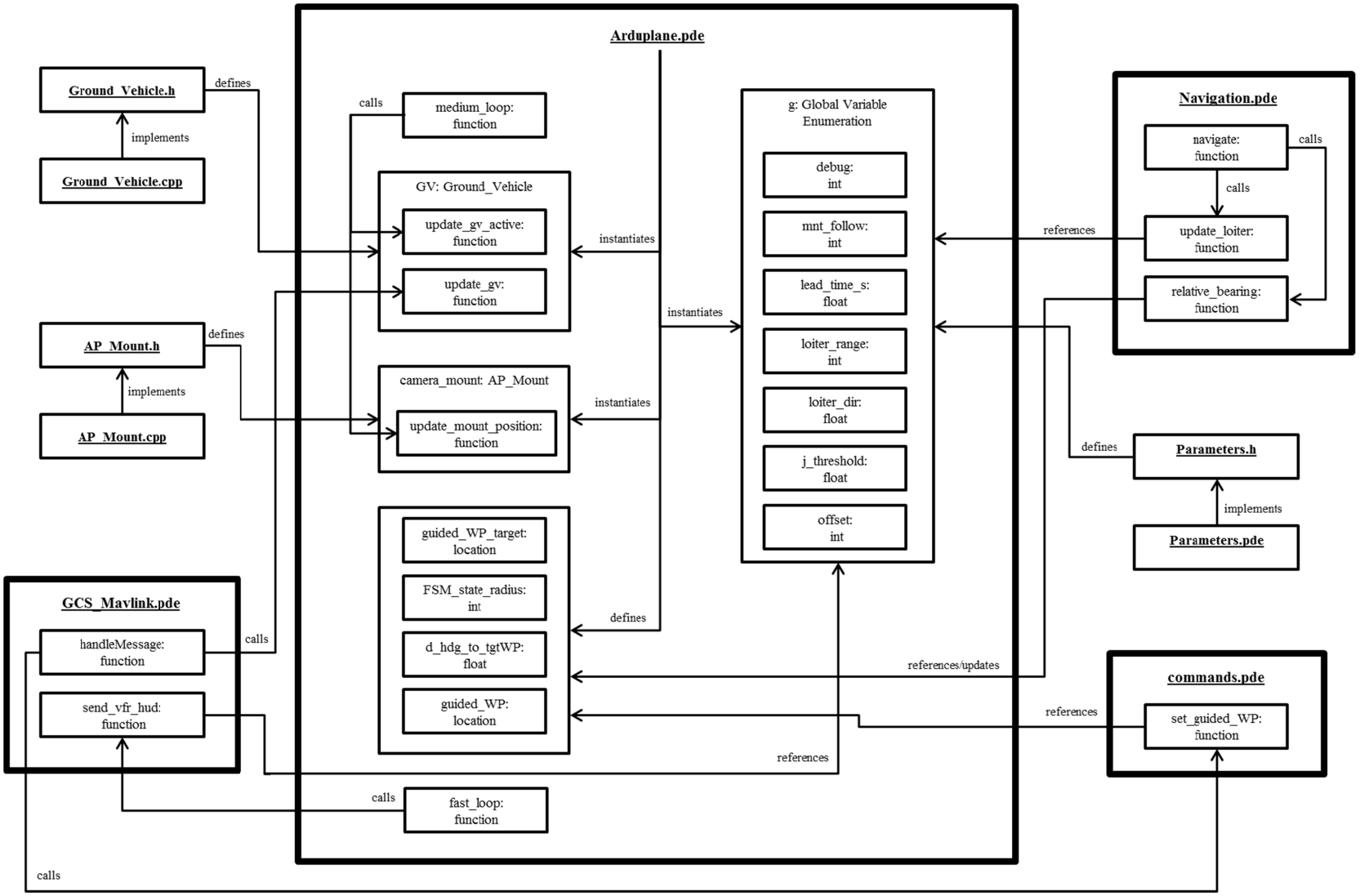

Figure 45: Diagram of Modified ArduPlane File Relationships

Appendix I: Real-World Rascal APM Parameters

```
AA_J_THRESHOLD,0.04
AA_LOITER_DIR,1
AA_LOITER_RANGE,20
AA_MNT_FOLLOW,2
AA_THRT_RATIO,0.75
AAA_DEBUG,0
AHRS_BARO_USE,0
AHRS_GPS_GAIN,1
AHRS_GPS_USE,1
AHRS_RP_P,0.3
AHRS_TRIM_X,-0.017
AHRS_TRIM_Y,0.076
AHRS_TRIM_Z,0
AHRS_WIND_MAX,0
AHRS_YAW_P,0.3
ALT_CTRL_ALG,0
ALT_HOLD_FBWCM,0
ALT_HOLD_RTL,10000
ALT_MIX,1
ALT_OFFSET,0
ALT2PTCH_D,0.2
ALT2PTCH_I,0.2
ALT2PTCH_IMAX,600
ALT2PTCH_P,1.75
AMP_OFFSET,0
AMP_PER_VOLT,27.32
ARSP2PTCH_D,0
ARSP2PTCH_I,0.1
ARSP2PTCH_IMAX,500
ARSP2PTCH_P,0.65
ARSPD_ENABLE,1
ARSPD_FBW_MAX,22
ARSPD_FBW_MIN,6
ARSPD_OFFSET,3517.628
ARSPD_RATIO,1.994
ARSPD_USE,0
BATT_CAPACITY,1760
BATT_CURR_PIN,-1
BATT_MONITOR,0
BATT_VOLT_PIN,-1
CAM_TRIGG_TYPE,0
CMD_INDEX,0
CMD_TOTAL,2
COMPASS_AUTODEC,1
COMPASS_DEC,-0.099
COMPASS_LEARN,1
COMPASS_OFS_X,-57.363
COMPASS_OFS_Y,-8.322
COMPASS_OFS_Z,85.877
COMPASS_USE,1
ELEVON_CH1_REV,0
ELEVON_CH2_REV,0
ELEVON_MIXING,0
ELEVON_REVERSE,0
ENRGY2THR_D,0
ENRGY2THR_I,0
ENRGY2THR_IMAX,20
ENRGY2THR_P,1
FBWB_ELEV_REV,0
FENCE_ACTION,0
FENCE_CHANNEL,0
FENCE_MAXALT,0
FENCE_MINALT,0
FENCE_TOTAL,0
FLAP_1_PERCNT,0
FLAP_1_SPEED,0
FLAP_2_PERCNT,0
FLAP_2_SPEED,0
FLTMODE_CH,8
FLTMODE1,10
FLTMODE2,2
FLTMODE3,2
FLTMODE4,2
FLTMODE5,0
FLTMODE6,0
FORMAT_VERSION,13
FS_GCS_ENABL,0
FS_LONG_ACTN,0
FS_SHORT_ACTN,0
GND_ABS_PRESS,98367.6
GND_TEMP,27.274
HDNG2RLL_D,0.1
HDNG2RLL_I,0.15
HDNG2RLL_IMAX,600
HDNG2RLL_P,1.5
INPUT_VOLTS,4.68
INS_ACCOFFS_X,1.227
INS_ACCOFFS_Y,-7.232
INS_ACCOFFS_Z,4.337
INS_ACCSCAL_X,1
INS_ACCSCAL_Y,1
INS_ACCSCAL_Z,1
INS_GYROFFS_X,-0.012
INS_GYROFFS_Y,0.033
INS_GYROFFS_Z,0.029
INS_MPU6K_FILTER,0
INS_PRODUCT_ID,88
INVERTEDFLT_CH,0
KFF_PTCH2THR,0
KFF_PTCHCOMP,0.125
KFF_RDDRMIX,0.4
KFF_THR2PTCH,0
LAND_FLARE_ALT,3
LAND_FLARE_SEC,2
LAND_PITCH_CD,0
LIM_PITCH_MAX,2500
LIM_PITCH_MIN,-2500
LIM_ROLL_CD,4000
LOG_BITMASK,0
MAG_ENABLE,1
MANUAL_LEVEL,0
MIN_GNDSPD_CM,0
MNT_ANGMAX_PAN,17999
MNT_ANGMAX_ROL,4500
MNT_ANGMAX_TIL,1000
MNT_ANGMIN_PAN,-18000
MNT_ANGMIN_ROL,-4500
MNT_ANGMIN_TIL,-9000
MNT_CONTROL_X,0
MNT_CONTROL_Y,0
MNT_CONTROL_Z,0
MNT_JSTICK_SPD,0
MNT_MODE,1
MNT_NEUTRAL_X,0
MNT_NEUTRAL_Y,-2200
MNT_NEUTRAL_Z,12
MNT_RC_IN_PAN,0
MNT_RC_IN_ROLL,0
MNT_RC_IN_TILT,0
MNT_RETRACT_X,0
MNT_RETRACT_Y,0
MNT_RETRACT_Z,0
```

```
MNT_STAB_PAN,1
MNT_STAB_ROLL,1
MNT_STAB_TILT,1
PTCH2SRV_D,0.23
PTCH2SRV_I,0.25
PTCH2SRV_IMAX,700
PTCH2SRV_P,2.3
RC1_DZ,30
RC1_MAX,1861
RC1_MIN,1143
RC1_REV,1
RC1_TRIM,1200
RC10_DZ,0
RC10_FUNCTION,0
RC10_MAX,1900
RC10_MIN,1100
RC10_REV,1
RC10_TRIM,1500
RC11_DZ,0
RC11_FUNCTION,0
RC11_MAX,1900
RC11_MIN,1100
RC11_REV,1
RC11_TRIM,1500
RC2_DZ,30
RC2_MAX,2014
RC2_MIN,990
RC2_REV,-1
RC2_TRIM,1200
RC3_DZ,3
RC3_MAX,1939
RC3_MIN,989
RC3_REV,1
RC3_TRIM,990
RC4_DZ,30
RC4_MAX,2015
RC4_MIN,989
RC4_REV,1
RC4_TRIM,1200
RC5_DZ,0
RC5_FUNCTION,6
RC5_MAX,1625
RC5_MIN,1390
RC5_REV,-1
RC5_TRIM,1500
RC6_DZ,0
RC6_FUNCTION,7
RC6_MAX,2000
RC6_MIN,1000
RC6_REV,1
RC6_TRIM,1500
RC7_DZ,0
RC7_FUNCTION,0
RC7_MAX,1499
RC7_MIN,1498
RC7_REV,1
RC7_TRIM,1499
RC8_DZ,0
RC8_FUNCTION,0
RC8_MAX,1761
RC8_MIN,989
RC8_REV,1
RC8_TRIM,1758
RC9_DZ,0
RC9_FUNCTION,0
RC9_MAX,1900
RC9_MIN,1100
RC9_REV,1
RC9_TRIM,1500
RLL2SRV_D,0.2
RLL2SRV_I,0.1
RLL2SRV_IMAX,500
RLL2SRV_P,2
RSSI_PIN,-1
RST_MISSION_CH,0
RST_SWITCH_CH,0
RUDDER_STEER,0
SCALING_SPEED,15
SERIAL3_BAUD,57
SR0_EXT_STAT,2
SR0_EXTRA1,10
SR0_EXTRA2,10
SR0_EXTRA3,2
SR0_PARAMS,50
SR0_POSITION,3
SR0_RAW_CTRL,1
SR0_RAW_SENS,2
SR0_RC_CHAN,2
SR3_EXT_STAT,1
SR3_EXTRA1,1
SR3_EXTRA2,1
SR3_EXTRA3,1
SR3_PARAMS,50
SR3_POSITION,1
SR3_RAW_CTRL,1
SR3_RAW_SENS,1
SR3_RC_CHAN,1
STICK_MIXING,1
SYS_NUM_RESETS,13
SYSID_MYGCS,255
SYSID_SW_TYPE,0
SYSID_THISMAV,1
TELEM_DELAY,0
THR_FAILSAFE,1
THR_FS_VALUE,950
THR_MAX,100
THR_MIN,0
THR_PASS_STAB,0
THR_SLEWRATE,35
THR_SUPP_MAN,0
THROTTLE_NUDGE,1
TRIM_ARSPD_CM,1200
TRIM_AUTO,0
TRIM_PITCH_CD,0
TRIM_THROTTLE,65
VOLT_DIVIDER,3.56
WHEELSTEER_D,0
WHEELSTEER_I,0
WHEELSTEER_IMAX,0
WHEELSTEER_P,0
WP_LOITER_RAD,150
WP_RADIUS,40
XTRK_ANGLE_CD,4500
XTRK_GAIN_SC,80
XTRK_MIN_DIST,50
XTRK_USE_WIND,1
YW2SRV_D,0.1
YW2SRV_I,0
YW2SRV_IMAX,0
YW2SRV_P,1.5
```

Appendix J: Simulated Rascal APM Parameters

```
AA_J_THRESHOLD,0.003
AA_LEAD_TIME_S,3
AA_LOITER_DIR,1
AA_LOITER_RANGE,65
AA_MNT_FOLLOW,2
AA_OFFSET_IN2OUT,35
AA_OFFSET_OUT2IN,35
AAA_DEBUG,0
AHRS_BARO_USE,0
AHRS_GPS_GAIN,1
AHRS_GPS_USE,1
AHRS_RP_P,0.4
AHRS_TRIM_X,0
AHRS_TRIM_Y,0
AHRS_TRIM_Z,0
AHRS_WIND_MAX,0
AHRS_YAW_P,0.4
ALT_CTRL_ALG,0
ALT_HOLD_FBWCM,0
ALT_HOLD_RTL,10000
ALT_MIX,1
ALT_OFFSET,0
ALT2PTCH_D,0
ALT2PTCH_I,0.1
ALT2PTCH_IMAX,500
ALT2PTCH_P,0.65
AMP_OFFSET,0
AMP_PER_VOLT,27.32
ARSP2PTCH_D,0
ARSP2PTCH_I,0.1
ARSP2PTCH_IMAX,500
ARSP2PTCH_P,0.65
ARSPD_ENABLE,0
ARSPD_FBW_MAX,22
ARSPD_FBW_MIN,6
ARSPD_OFFSET,1120.364
ARSPD_RATIO,1.994
ARSPD_USE,0
BATT_CAPACITY,1760
BATT_CURR_PIN,2
BATT_MONITOR,0
BATT_VOLT_PIN,1
CAM_TRIGG_TYPE,0
CMD_INDEX,0
CMD_TOTAL,0
COMPASS_AUTODEC,1
COMPASS_DEC,-0.071
COMPASS_LEARN,1
COMPASS_OFS_X,4.372
COMPASS_OFS_Y,12.571
COMPASS_OFS_Z,-17.435
COMPASS_USE,1
ELEVON_CH1_REV,0
ELEVON_CH2_REV,0
ELEVON_MIXING,0
ELEVON_REVERSE,0
ENRGY2THR_D,0
ENRGY2THR_I,0
ENRGY2THR_IMAX,20
ENRGY2THR_P,0.5
FBWB_ELEV_REV,0
FENCE_ACTION,0
FENCE_CHANNEL,0
FENCE_MAXALT,0
FENCE_MINALT,0
FENCE_TOTAL,0
FLAP_1_PERCNT,0
FLAP_1_SPEED,0
FLAP_2_PERCNT,0
FLAP_2_SPEED,0
FLTMODE_CH,8
FLTMODE1,10
FLTMODE2,11
FLTMODE3,5
FLTMODE4,2
FLTMODE5,2
FLTMODE6,0
FORMAT_VERSION,13
FS_GCS_ENABL,0
FS_LONG_ACTN,0
FS_SHORT_ACTN,0
GND_ABS_PRESS,97488.42
GND_TEMP,32.23528
HDNG2RLL_D,0.1
HDNG2RLL_I,0.02
HDNG2RLL_IMAX,500
HDNG2RLL_P,1
INPUT_VOLTS,4.68
INS_ACCOFFS_X,27.988
INS_ACCOFFS_Y,-0.098
INS_ACCOFFS_Z,-82.307
INS_ACCSCAL_X,1
INS_ACCSCAL_Y,1
INS_ACCSCAL_Z,1
INS_GYROFFS_X,0
INS_GYROFFS_Y,0
INS_GYROFFS_Z,0
INS_MPU6K_FILTER,0
INS_PRODUCT_ID,0
INVERTEDFLT_CH,0
KFF_PTCH2THR,0
KFF_PTCHCOMP,0.35
KFF_RDDRMIX,0.25
KFF_THR2PTCH,0
LAND_FLARE_ALT,3
LAND_FLARE_SEC,2
LAND_PITCH_CD,0
LIM_PITCH_MAX,2000
LIM_PITCH_MIN,-2000
LIM_ROLL_CD,4500
LOG_BITMASK,334
MAG_ENABLE,1
MANUAL_LEVEL,0
MIN_GNDSPD_CM,0
MNT_ANGMAX_PAN,17999
MNT_ANGMAX_ROL,4500
MNT_ANGMAX_TIL,8000
MNT_ANGMIN_PAN,-17999
MNT_ANGMIN_ROL,-4500
MNT_ANGMIN_TIL,-8000
MNT_CONTROL_X,0
MNT_CONTROL_Y,-40
MNT_CONTROL_Z,90
MNT_JSTICK_SPD,0
MNT_MODE,1
MNT_NEUTRAL_X,0
MNT_NEUTRAL_Y,0
MNT_NEUTRAL_Z,0
MNT_RC_IN_PAN,0
MNT_RC_IN_ROLL,0
MNT_RC_IN_TILT,0
MNT_RETRACT_X,0
```

```
MNT_RETRACT_Y,0
MNT_RETRACT_Z,0
MNT_STAB_PAN,1
MNT_STAB_ROLL,0
MNT_STAB_TILT,1
PTCH2SRV_D,0.15
PTCH2SRV_I,0.2
PTCH2SRV_IMAX,700
PTCH2SRV_P,2
RC1_DZ,30
RC1_MAX,1911
RC1_MIN,1096
RC1_REV,-1
RC1_TRIM,1200
RC10_DZ,0
RC10_FUNCTION,0
RC10_MAX,1900
RC10_MIN,1100
RC10_REV,1
RC10_TRIM,1500
RC11_DZ,0
RC11_FUNCTION,0
RC11_MAX,1900
RC11_MIN,1100
RC11_REV,1
RC11_TRIM,1500
RC2_DZ,30
RC2_MAX,1903
RC2_MIN,1092
RC2_REV,-1
RC2_TRIM,1200
RC3_DZ,3
RC3_MAX,1900
RC3_MIN,1085
RC3_REV,1
RC3_TRIM,1086
RC4_DZ,30
RC4_MAX,1898
RC4_MIN,1086
RC4_REV,-1
RC4_TRIM,1200
RC5_DZ,0
RC5_FUNCTION,7
RC5_MAX,2000
RC5_MIN,1000
RC5_REV,1
RC5_TRIM,1552
RC6_DZ,0
RC6_FUNCTION,6
RC6_MAX,2000
RC6_MIN,1000
RC6_REV,1
RC6_TRIM,1498
RC7_DZ,0
RC7_FUNCTION,0
RC7_MAX,1498
RC7_MIN,1497
RC7_REV,1
RC7_TRIM,1498
RC8_DZ,0
RC8_FUNCTION,10
RC8_MAX,1900
RC8_MIN,1100
RC8_REV,1
RC8_TRIM,1901
RC9_DZ,0
RC9_FUNCTION,0
RC9_MAX,1900
RC9_MIN,1100
RC9_REV,1
RC9_TRIM,1500
RLL2SRV_D,0.08
RLL2SRV_I,0.2
RLL2SRV_IMAX,1000
RLL2SRV_P,1.75
RSSI_PIN,-1
RST_MISSION_CH,0
RST_SWITCH_CH,0
RUDDER_STEER,0
SCALING_SPEED,15
SERIAL3_BAUD,57
SR0_EXT_STAT,2
SR0_EXTRA1,10
SR0_EXTRA2,10
SR0_EXTRA3,2
SR0_PARAMS,50
SR0_POSITION,3
SR0_RAW_CTRL,50
SR0_RAW_SENS,2
SR0_RC_CHAN,2
SR3_EXT_STAT,0
SR3_EXTRA1,0
SR3_EXTRA2,0
SR3_EXTRA3,0
SR3_PARAMS,0
SR3_POSITION,0
SR3_RAW_CTRL,0
SR3_RAW_SENS,0
SR3_RC_CHAN,0
STICK_MIXING,1
SYS_NUM_RESETS,26
SYSID_MYGCS,255
SYSID_SW_TYPE,0
SYSID_THISMAV,1
TELEM_DELAY,0
THR_FAILSAFE,1
THR_FS_VALUE,950
THR_MAX,100
THR_MIN,0
THR_PASS_STAB,0
THR_SLEWRATE,20
THR_SUPP_MAN,0
THROTTLE_NUDGE,1
TRIM_ARSPD_CM,2500
TRIM_AUTO,0
TRIM_PITCH_CD,0
TRIM_THROTTLE,65
VOLT_DIVIDER,3.56
WHEELSTEER_D,0
WHEELSTEER_I,0
WHEELSTEER_IMAX,0
WHEELSTEER_P,0
WP_LOITER_RAD,150
WP_RADIUS,45
XTRK_ANGLE_CD,5500
XTRK_GAIN_SC,60
XTRK_MIN_DIST,50
XTRK_USE_WIND,1
YW2SRV_D,0.7
YW2SRV_I,0.01
YW2SRV_IMAX,0
YW2SRV_P,0.75
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
27-03-2014	Master's Thesis	Aug 2012 - Mar 2014

4. TITLE AND SUBTITLE

Feasibility of Onboard Processing of Heuristic Path Planning and Navigation Algorithms within SUAS Autopilot Computational Constraints

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Neal, Charles J, Capt

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENV-14-M-44

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RQQA
2210 8th Street
Bldg 146, Room 300
Wright-Patterson AFB, OH 45433
COMM 937-713-7038; Email: derek.kingston@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Statement A
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research addresses the flight path optimality of Small Unmanned Aerial Systems (SUAS) conducting overwatch missions for convoys or other moving ground targets. Optimal path planning algorithms have been proposed, but are computationally excessive for real-time execution. Using the Arduino-based ArduPilot Mega Unmanned Aerial Vehicle (UAV) autopilot system, Hardware-in-the-Loop (HIL) analysis is conducted on default mobile target tracking methods. Designed experimentation is used to determine autopilot settings that improve performance with respect to path optimality. Optimality is characterized using a weighted combination of stand-off range and aircraft roll-rate. Finally, a state-based heuristic navigation strategy is designed, developed, and tested that approximates optimal path solutions and can be used for real-time execution. A 66% improvement in mean performance is achieved over default target tracking methods. Finite state machine improvements are found to be statistically significant and it is concluded that heuristic strategies can be a viable approach to realizing near-optimal SUAS flight paths utilizing onboard processing capabilities.

15. SUBJECT TERMS

UAV, SUAS, autopilot, optimal path planning, mobile target tracking, convoy overwatch, heuristic approximation, finite state machine, hardware-in-the-loop, flight test

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. John M. Colombi AFIT/ENV
U	U	U	UU	116	19b. TELEPHONE NUMBER (Include area code) (937) 255-3636 x3347 john.colombi@afit.edu